TR-1048

AGILE CONTROL OF MILITARY OPERATIONS (JFACC)

David Logan (PM)
Jerry M. Wohletz, Ph.D. (PL)
David A. Castañon, Ph.D.
Michael Curry
Brain Bank

ALPHATECH, Inc.
50 Mall Road
Burlington, MA 01803
(781) 273-3388

NOVEMBER 2001
CDRL ITEM A004

FINAL REPORT

INFORMATION SYSTEMS DIVISION
INFORMATION DIRECTORATE
AIR FORCE RESEARCH LABORATORY
26 ELECTRONIC PARKWAY
ROME, NY 13440-4514

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATED COVERED
	November 2001	Experimental Report, 08/27/1999-10/31/2001

4. TITLE AND SUBTITLE
AGILE CONTROL OF MILITARY OPERATIONS (JFACC)

5. FUNDING NUMBERS
F30602-99-C-0203

6. AUTHOR(S)
Logan, D.A., Wohletz, J.M., Castañon, D.A., Curry, M.L., Bank, B.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ALPHATECH, Inc.
50 Mall Road
Burlington, MA 01803

8. PERFORMING ORGANIZATION REPORT NUMBER
TR-1048

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/IFKRD
26 Electronic Parkway
Rome, NY 13440-4514
ATTN: Carl Defranco, AFRL/IFSA

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; Distribution Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This research focused on the problem of providing military commanders with real-time, optimal control of military air-to-ground operations for a 24-hour segment of a Joint Air Operations (JAO) campaign. In particular, we focused on developing control algorithms that anticipate possible air-to-ground mission modifications due to uncertain future events, thereby generating missions that can be readily adapted in the presence of contingencies. The primary benefit of this technology is agile and stable control of distributed and dynamic military operations conducted in inherently uncertain, hostile, and rapidly changing environments. The control methodology developed combines Approximate Dynamic Programming (ADP) and statistical hybrid state modeling techniques. Accordingly, a novel hybrid, multi-rate control architecture that tailors the control strategy for different battlespace situations was developed. For this JAO problem, a key concern was the scalability of the control methodology to larger scenarios. As a result, we investigated a broad spectrum of ADP control strategies. The solution quality and computation performance of these algorithms was tested and verified in a JAO simulator. It was shown through experimentation that the ADP strategies were able to produce operationally consistent control strategies that anticipated likely contingencies and positioned assets for opportunities of recourse in near real-time.

14. SUBJECT TERMS

15. NUMBER OF PAGES
112

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

1 EXECUTIVE SUMMARY

The current process for planning of missions in Joint Air Operations (JAO) is slow and manually intensive. This research, performed by ALPHATECH, focused on the problem of providing military commanders with real-time, optimal control of military air-to-ground operations through the use of fast, near optimal mission replanning, using control algorithms that anticipate possible mission modifications due to uncertain future events for a 24-hour segment of a JAO campaign. The primary benefit of this technology is agile and stable control of distributed, dynamic military operations conducted in inherently uncertain, hostile, and rapidly changing environments. The goal of the JAO controller is to achieve specified Joint Force Air Component Commander (JFACC) objectives while minimizing the friendly asset losses. The controller generates or updates mission definitions for both base assets and airborne assets. The mission definition includes a target, mission waypoints, strike package composition, weapon composition, and desired time-on-target. Key features of the JAO environment include risk and reward that are dependent on package composition. The outcome of JAO events — target destruction, threat destruction, friendly asset attrition, emerging targets and threats — are uncertain, and thus missions must be adapted based on the observed outcomes. Due to the potential scarcity of resources, efficient resource utilization and adaptation to emerging battlespace conditions are paramount to assure a successful mission.

In order to develop controllers which address the uncertain, dynamic nature of the JAO statement, we adopted a framework for dynamic decision making under uncertainty, known as Markov Decision problems. In this framework, optimal decisions are chosen based on the most recent information, and the selected decisions must hedge against possible future contingencies. This explicit modeling of future contingencies results in *proactive* versus *reactive* control behaviors. This proactive attribute is desirable for stable and agile control of the JAO enterprise because future information arrival and control opportunities are dependent on stringent spatial, temporal, and coordination constraints.

The principal approach for control design using Markov Decision problems is the Stochastic Dynamic Programming (SDP) algorithm. However, it is well known that this approach suffers from the *curse of dimensionality* and is intractable (lacks scalability) for realistic sized problems. Thus, our investigations focused on developing Approximate Dynamics Programming (ADP) strategies that provide the desirable *proactive* control behaviors but can be

computed in near real-time for realistic JAO scenarios. The control design technology is based on combining hybrid state modeling techniques for developing statistical dynamical models relating mission decisions to evolution of objects in the battlespace, together with ADP control design techniques that have demonstrated real-time, proactive performance for other relevant military problems. In our investigations, we developed a broad spectrum of ADP control techniques for the JAO problem, and evaluated their relative performance and scalability. The technical accomplishments of this research are summarized below:

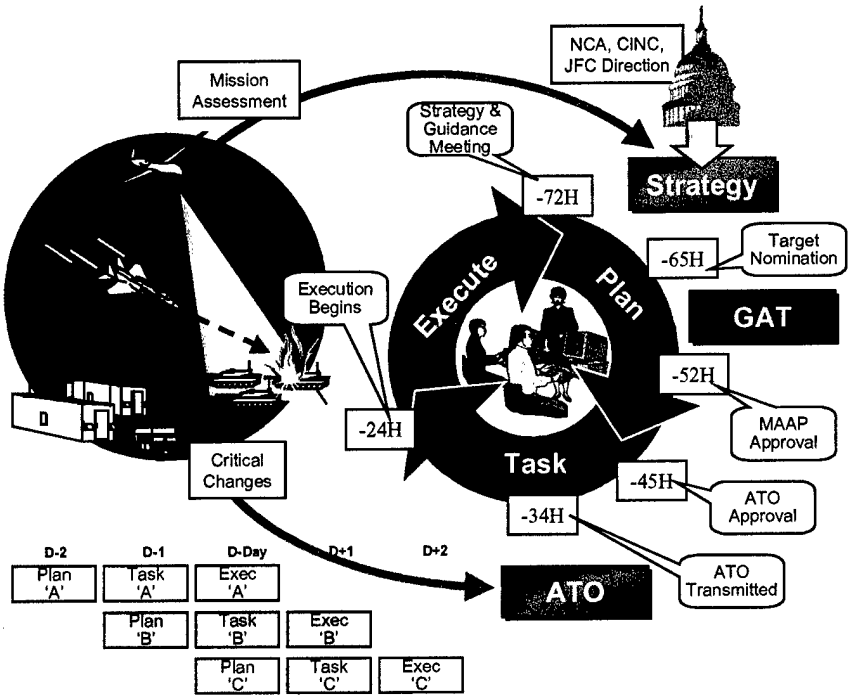
- Translated the JAO Control Enterprise into a Dynamical Hybrid State, Discrete Event, Stochastic Decision Making Problem
- Integrated Emerging ADP Technologies into JAO Feedback Controllers
- Experimentally Demonstrated the Benefits of Feedback Control
- Experimentally Demonstrated Benefits of Approximate Optimal Control
- Developed Innovative Hybrid, Multi-Rate Control Architecture
- Developed Computationally Efficient Control Algorithms that Produce Operationally Consistent Behaviors
- Extended Control Algorithms to Accommodate Hierarchical Mission Tasking and ISR Information Collection

The control algorithms developed in these investigations achieve the desired research objective of automating military operations planning to provide real-time, near-optimal control strategies that achieve operational objectives while minimizing asset losses. Adopting a hybrid, multi-rate control architecture permitted the tailored application of control to the operational situation at hand; faster control was used when actions had relatively local effects (e.g. a mission detour or divert), followed by slower modifications to address overall mission strategy. Given this architecture, a spectrum of ADP control strategies were developed and implemented that produce immediate restasking or abort decisions to preplanned multiple wave tasking. The solution quality and computation performance of these algorithms was tested and verified in a JAO simulator. It was shown through experimentation that the ADP strategies were able to produce operationally consistent, proactive control strategies that anticipated likely contingencies and positioned assets for opportunities of recourse all in either real-time or near real-time. Furthermore, a scalability assessment illustrated that many of the ADP controllers could provide near real-time performance for scenarios with 250 targets with some modest parallel computatio.

2 INTRODUCTION

2.1 BACKGROUND

The current process for JAO planning and execution—based on the steps of Strategy Development, GAT (Guidance, Apportionment, and Targeting), MAAP (Master Air Attack Planning), and Air Tasking Order (ATO) Production—is illustrated in Figure 1. The product is published every 24 hours. End-to-end development time typically requires 72 hours.



The ATO Process is a Continuous, Overlapped 72 Hour Process.

The JFACC Strategy team develops the overarching Air & Space Strategy in concert with the JFC's Operations Plan. They develop, refine, and disseminate the long-range Theater Air & Space Operations Plan. They continually assess plan execution against their overarching strategy.

The JFACC GAT team develops 1) the Guidance Letter that addresses planning and apportionment, 2) the Joint Integrated Prioritized Target List (JIPTL) accordance with prioritized tasks in the Air & Space Ops Plan, and 3) the Master Air Attack Plan (MAAP) that contains JFC & JFACC guidance, support plans, component requests, target update requests, forces availability, target information, aircraft allocation, etc.

The JFACC ATO Production team prepares detailed plans that provide operational and tactical direction to wing level commanders including: 1) the detailed Air Tasking Order (ATO), 2) Special Instructions (SPINS), and 3) the Airspace Coordination Orders (ACO).

Figure 1 Key Steps in the Current JAO Process Limit Agility and Responsiveness

The heavily sequential nature of the current JAO process hinders the JFACC's ability to operate within the decision cycles of our adversaries. Moreover, brute force attempts to adapt the current process (e.g., diverting resources to engage time-critical targets) often lead to unstable operations. Given the realities of the dynamic problem above, the current process suffers from the following limitations.

Lack of Agility: Today's JAO planning tools tend to produce tightly woven plans. Given the problem as stated by the user, planners recognize the dependencies between resources, and generate solutions that maximize effectiveness by pushing resources to limits. Alas, unanticipated changes to the plan can generate significant ripple effects due to strong dependencies. These dependencies are intrinsic to the JAO control problem: the delivery of a

missile to a target requires the coordination of multiple resources 1) to locate and identify the target, 2) to launch the weapon, 3) to provide safe ingress and egress, 4) to provide sufficient fuel to airborne assets and 5) to assess bomb damage.

Planners attempt to avoid ripple effects by: 1) scheduling redundancy into the plan; or when changes are needed, by 2) terminating execution of all portions of the plan related to the breakpoint, or 3) ignoring the dependencies and allow the other portions of the plan to continue. Redundancy implies inefficient use of assets, termination of a portion of a plan implies ineffective use of assets, and ignoring dependencies implies risk of instability, i.e., mission failure.

Ad Hoc Stability: Current systems rely on human operators to serve as a stabilizing force by assessing the likely impact of unanticipated events. Although human operators are *very* good at adapting to new situations, their performance degrades dramatically when they are overloaded with information and tasks. Since the effectiveness of the assessment depends vitally on the operator's insight into the plan, and on the time available to make a decision, the drawback to this approach is the highly subjective and non-comprehensive nature of the operator's assessment. In addition, this task often becomes a pacing task, preventing the approach from scaling to highly dynamic environments.

Ineffective Feedback: Current systems are ineffective at providing feedback to guide the use of JAO resources. Fortunately, more sensors will soon supply more timely information on the state of the battlespace. Given sufficient agility in the attack and sensor platforms, the challenge is to reengineer the JAO process to incorporate this information in a disciplined manner. This also requires the JAO planning process to actively guide the information collection process in support of mission execution by providing timely information need specifications.

Ineffective Use of Assets and Resources: Current systems tend to build redundancy into the plans in order to provide a degree of agility (e.g., place aircraft on "SCUD CAP" in case a time-critical target (TCT) appears). Newer concepts focus on diverting ongoing missions to deal with significant changes in the situation. Unfortunately, this agility often comes at the expense of major disruptions to the execution of the remainder of the plan. For example, the diversion of an electronic countermeasures mission to support a TCT kill mission may result in the cancellation of several planned missions—again, leading to ineffective use of our resources.

2.2 PROBLEM DESCRIPTION

Within the Joint Air Operations Enterprise model, ALPHATECH's research is focused on generating real-time, optimal control strategies for a 24-hour segment on a JAO campaign. It is assumed that some form of higher level decomposition of the battle space has been performed, and that an Area of Responsibility (AOR) has been defined that includes approximately 100 targets. The goal of the controller is to achieve the specified JFACC objective for the AOR while minimizing the friendly asset losses. The controller will generate/update mission definitions for both base assets and airborne. The mission definition includes a target, high-level waypoints, strike package composition, weapon composition, and desired time-on-target. Inputs to the controller include a known target list, available assets, known threats list, and some indication of the likelihood of emerging target and/or threats. Through the continuous gathering of Intelligence, Surveillance, and Reconnaissance (ISR) data, the control system will generate these mission definitions on a time scale of approximately 15 minutes. The updates can be as benign as continue current plan or as drastic as reroll some packages, abort others, or launch new packages.

Key features of the JAO environment of interest include risk and reward that are dependent on package composition and weaponeering. The outcome of JAO events — target destruction, threat destruction, friendly asset attrition, emerging targets and threats — are uncertain in realistic JAO environments. Finally, limited resources and dealing with emerging and threats are paramount to the successful execution of the JFACC objective.

2.3 THEORETICAL TECHNIQUE

To address the limitations of the current JAO control process, ALPHATECH will investigate the utility of a comprehensive new approach based upon modern control theory. Because of its complexity and the time critical nature of the decisions that must be made, automated decision aids must support JAO operators. Past attempts at developing such decision aids have been based upon planning technology. Unfortunately, the plans produced by these decision aids are often rendered obsolete by unforeseen events. This difficulty can be addressed by periodic replanning, either in full or (more commonly) by partial modification of the plan.

However, planning technology provides little guidance on 1) how this replanning should be conducted, nor 2) how plans can be made robust to the need for replanning and repair.

In contrast, control theory uses the idea of continuous feedback to minimize the impact of uncertainty. Uncertainty is explicitly represented in both dynamics and observation models. Feedback control laws select current decisions as a function of the current (estimated) state of the system. These control laws are selected to optimize a quantitative objective criterion, subject to constraints on controls, dynamics, and observability. Control laws for current decisions explicitly consider the fact that future decisions will be made in the same optimal manner, based upon state information available in the future [B96]. Thus control theory provides an extensive conceptual foundation for continuous JAO.

2.4 VALUE TO MILITARY

The goal of this research is to provide real-time dynamic control of military air operations via near optimal mission assignments, which are robust under replanning. The primary benefit of this technology is agile and stable control of distributed and dynamic military operations conducted in inherently uncertain, hostile, and rapidly changing environments.

Utilizing the available real-time information from the battle space, the propose algorithm generates mission assignments resembling an ATO to achieve a desired JFACC objective. The fact of using feedback to create or update mission assignment desensitizes the desired outcome to modeling error and uncertainties that are inherently associated with military air operations. Furthermore, by formulating the control problem as an optimal control problem, the recommendations will anticipate key uncertainties and provide opportunities for recourse. In fact, the optimal solution will achieve the desired JFACC objective by minimizing the operational cost and risk to our assets. As an example, the optimal solution can identify a critical communication linkage that if destroyed can achieve air superiority without having to destroy every component of an Integrated Air Defense System (IADS). Given the progression towards more Unmanned Air Vehicles (AUVs), the proposed system could be used to automatically provide the UAV with its mission. The benefits of this technology to the military is summarized below using the taxonomy presented in the BAA:

Increased Agility: Approximate optimal control techniques generate solution that permit opportunities of recourse for key uncertain JAO outcomes; thus, increasing the agility of JAO operations by proactively (versus reactively), consistently, and efficiently responding to changes in the environment.

Flexibility: This technology is applicable to a wide spectrum of military conflicts.

Stability: Feedback control provides an automated system to stabilize the JAO environment; thus, eliminating the need for ad hoc stabilization via human operators.

Effective Feedback: Feedback control provides an natural framework to fuse large volumes of data to produce a coherent control strategies.

Effective Use of Assets and Resources: Optimal control generates control solution that effectively use the available resources.

Automated Operations: Envisioned is a prototype system that fits into existing C2 AOC. This system would monitor the progression of the battle space and generate real-time control strategies. The system could automatically manage autonomous assets.

2.5 REPORT LAYOUT

This report has three primary sections. In Section 3, the details of the JAO plant dynamics are presented. Next, Section 4 contains a detailed discussion of the JAO controllers formulation and implementation. Finally, Section 5 summarizes the key experimental results for the dynamics and controllers that were implemented. In addition to these primary sections, a series of appendices are included to compliment the main body of the report.

3 JAO PLANT DYANMICS

In this section, we develop the JAO plant dynamics in which air package composition and tasking are represented in the context of a Joint Force Air Component Commander (JFACC) air-to-ground probem. Our model provides a flexible representation of the JFACC problem that "matches" the fidelity of our control approach (and has evolved accordingly), allowing for efficient experimentation. The presentation of the JAO plant dynamics begins with a general discussion of the JAO air-to-ground problem. This discussion is followed by the presentation of the dynamics that were implemented for this research. Finally, the details of the JAO plant dynamics implementation will be presented.

3.1 JOA AIR-TO-GROUND PROBLEM DESCRIPTION

In this section, we describe the general JFACC air-to-ground problem from which we will distill the salient JAO plant dynamics in subsequent sections.

The objective of the JFACC strike mission planning problem is to maximize damage to enemy targets while minimizing losses of our own aircraft. Strike missions involve targets, air defense units, strike aircraft, and Suppression of Enemy Air Defense (SEAD) aircraft. We discuss the roles of each below.

Targets are objects the JFACC wishes to damage. The JFACC objectives may specify a desired effect such as destroying, temporarily disabling, or disrupting the target. Targets are vulnerable to strike aircraft, which are discussed below. Multiple strike aircraft or special tactics may be required to achieve the desired effect when targets are geographically dispersed (i.e., have multiple aim points). Targets may also be coupled (physically or logically), and the strike mission may need to achieve some threshold damage before attaining the JFACC's objective. Collateral damage is always a concern, and application of strike force above some threshold may lead to undesirable outcomes. Multiple strike missions may also need to be coordinated in time (i.e., sequenced or synchronized) to achieve the desired effects.

The true state of the target (including location, identity, function, and value) is dynamic and may not be known with certainty. Targets may move or hide, and their functions may change, making their prosecution more or less desirable over time. Strike mission planners do not necessarily know all potential targets beforehand, and they may elect to keep some strike forces in reserve to attack targets that were heretofore unknown. These dynamics imply that the targets vulnerability and value vary with time. A time is typically specified for which a specific

strike mission is likely to achieve the desired effect. In general, strike missions will need to be tailored depending on when and where the target will be prosecuted. Adversaries protect targets using a combination of passive and active defense measures. Passive measures make targets harder to kill and/or prosecute; they include hardening, cover, concealment, and deception (HCCD). Active defense directly attacks aircraft in the strike mission. We discuss Air Defense units next.

Air Defense (AD) units defend targets. The effectiveness of an AD unit depends on the type of AD unit, the type attacking aircraft, the tactics of both the aircraft and the AD unit, and a variety of external influences such as weather. AD units are vulnerable to various type of suppression, which may be available on the strike aircraft but are concentrated typically on SEAD aircraft (discussed below). Suppression may be in the form of munitions that destroy critical components of the AD unit. Although this is typically permanent, AD units may be repaired given enough time. Other types of suppression such as jamming, chaff, and decoys, temporally disable or deceive AD units, which make them less effective. The AD unit's effectiveness may be enhanced by strategic placement, such as protecting multiple targets or overlapping to protect a single target. AD units may also be connected or coupled which tend to improve their effectiveness and reduce their vulnerability.

Similar to targets, the true state of the AD units is dynamic and may not be known with certainty. The AD units may be known or unknown, move or hide, they may choose not to expose themselves to suppression aircraft, or may operate at a lower effectiveness to reduce vulnerability.

Strike aircraft prosecute targets. Prosecution encompasses a range of effects, from permanently destroying the target, to temporarily disabling it, to simply disrupting it's operation. However, strike aircraft are vulnerable to enemy Air Defense (AD). Therefore, the aircraft's effectiveness is judged by its ability to circumvent AD and achieve the desired effect on the target. To accomplish these objectives, the aircraft is configured with munitions and the ability to suppress enemy air defense.

Each Strike aircraft is configured with a limited number of munitions, which are non-renewable and may only be replenished at an airbase (provided the supply exists). Selection of the munitions, as well as the tactics used to deploy them, determine the effect on the target (i.e., destroy, disable, or disrupt) and depend on several external influences including terrain, type of

target, air defenses, potential for collateral damage, and weather. Much of the strike aircraft's ability to suppress enemy air defense also comes from nonrenewable resources such as chaff, flares, and decoys. Strike aircraft may also be equipped with self-protection jamming equipment; this equipment is not depletable, but has power and duty cycle constraints. We discuss renewable resources for suppressing enemy air defense further when we introduce Weasel aircraft below. Another important nonrenewable is resource is fuel. Fuel may be replenished at appropriate facilities, including air bases and orbiting tanker stations. SEAD aircraft help the strike aircraft through the enemy's air defenses. Some SEAD aircraft are configured to physically attack air defense units; these are Weasel aircraft. Other SEAD aircraft are configured to jam air defense units (primarily air defense radars); these are Jammers.

Weasel aircraft are configured with munitions; these are depletable resources. Jammers are equipped with an array of jamming pods; these are not depletable, but have power and duty-cycle constraints.

SEAD aircraft are packaged with strike aircraft to support a particular mission or placed on CAP to support a group of missions in a particular area. Jammer aircraft are slower than strike and weasel aircraft, but their effect may cover a wide geographical area. Therefore, these aircraft are ideal for supporting multiple strike missions in a particular geographical area.

To achieve the JFACC objectives, air packages, i.e. team of aircraft that coordinate their activities and fly in either a loose or tight formation, must be composed, tasked, and retasked such that the maximum number of targets are prosecuted with minimal resource losses. In general, air package configuration number and type of aircraft and resources. Resources include munitions type and quantity, SEAD capability (including self-protection pods), SEAD configuration (e.g., frequency range), and extra fuel tanks. Air package tasking includes target designation, course routing information, and potential opportunities for future retasking. Retasking involves updating an air package tasking in flight, which is limited by a variety of constraints but may be valuable to high value, fleeting targets.

In the following, we develop our hybrid dynamical model for Joint Air Operations (JAO). We start with a general discussion of hybrid dynamical models. Based on this general discussion, we develop hybrid dynamical models for each of the objects in the JAO environment. Finally, we introduce a composite hybrid dynamical model that facilitates interaction among JAO

objects. The hierarchical structure of the composite hybrid dynamical model simplifies design and analysis, as well as permitting a more lucid exposition of this complex environment.

3.2 HYBRID DYNAMICAL MODEL FORMULATION

Consider the following hybrid dynamical system based on [Bran95]

$$H = (Q, \Sigma, A, G, V, C, F)$$

where Q is a discrete state space. Each state (or mode) $q \in Q$ is associated with a controlled dynamical system, $\Sigma_q \in \Sigma$, an autonomous jump set $A_q \in A$, and a controlled jump set $C_q \in C$. The controlled dynamical system is given by $\Sigma_q = [X_q, \Gamma_q, \phi_q, U_q]$ where X_q is the state space, Γ_q is an appropriately defined timing map, ϕ_q represents the dynamics, and U_q is the continuous control set. If the state of the dynamical system enters $A_q \subset X_q$, the system jumps autonomously to some new mode $p \neq q$ according to a control mapping G_q, which is parameterized by the control set V_q. Alternatively, if the state enters $C_q \subset X_q$, the system may be instructed to jump to some new mode $p \neq q$ such that $X_p \subset F_q$.

3.2.1 Battlespace Objects

In this section, we develop the hybrid dynamical models for air packages, targets, and threats.

3.2.1.1 Air Packages

An air package dynamics will be using the hybrid dynamical representation in the above section. The discrete state

$$q \in \mathrm{Z}_+^2 \times \{Ingress, Egress, Base\}$$

where Z_+^2specifies the number of Strike and Weasel aircraft and $\{Ingress, Egress, Base\}$ identifies the current operating conditions of the air package; *Ingress* indicates that the air package is fully loaded and may be retasked, *Egress* indicates that the air package has already engaged its target and is not eligible for retask, and *Base* indicates that the air package's mission is completed and the associated aircraft are available for assembly into new air packages. The continuous dynamics of an air package are independent of the discrete state and are given by

$$\Sigma = [X, \Gamma, \phi, \mu(r)]$$

where $X \in \Re^2$ is the (x, y) location of the air package, Γ is an appropriately defined timing map, ϕ is a constant velocity kinematic model, and $\mu(r): X \rightarrow U$ is a flight controller that

maps the current state into a continuous control input $u \in U$ given the next waypoint r. Disturbances are neglected in the continuous dynamics. Waypoints may be introduced through either autonomous jumps via the control set V (e.g., last waypoint has been reach, pick-up the next waypoint or loiter) or through controlled jumps specified in C (e.g., retask or abort mission). We neglect disturbance with respect to waypoint selection, i.e., no navigational error.

The autonomous jump set A is similarly defined for all q, including waypoint arrival, which instigates subsequent actions. Autonomous jumps also represent various interactions when considered within a composite model; these jumps are summarized in Figure 2 and will be discussed in detail in Section 3.2.2 below. From this figure we see that *threat engagements* may diminish the aircraft in the air package, and a single engagement has the potential to destroy all aircraft; and *Target engagements* transition the air package from ingress to egress.

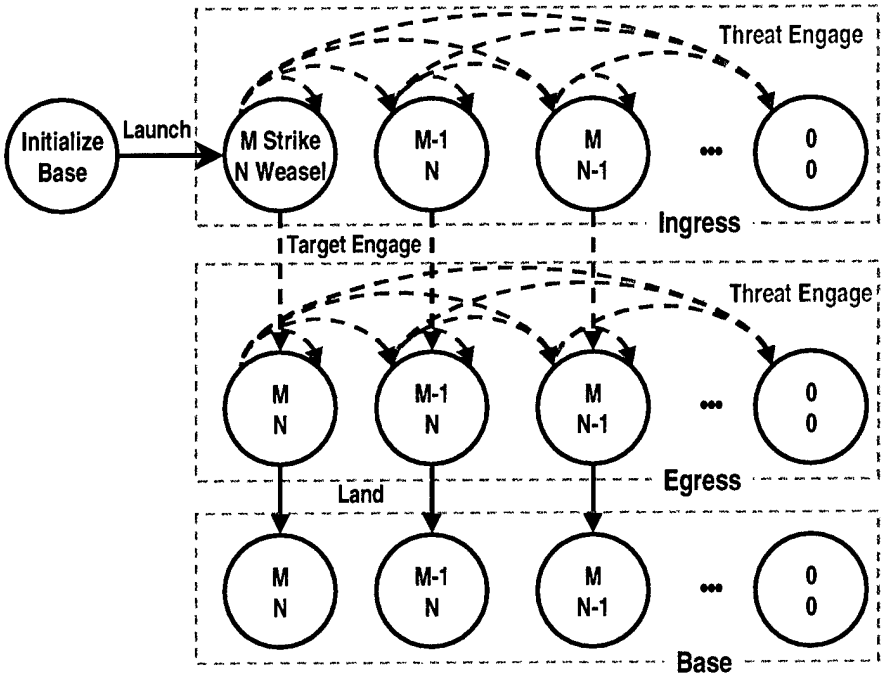


Figure 2 Autonomous Jumps Represent Interactions with Air Packages

The controlled jump set $C \subset X$ enables changes in tasking and/or configuration of air packages. The mapping $F: C \rightarrow 2^S$ where $S = X \times Q$ specifies a set of feasible jumps for each state $x \in C$ (e.g., reconfiguration at the base). We obtain the familiar control theoretic notation by recognizing that in the absence of uncertainty, $F(x)$ is the set of possible control actions. In a stochastic setting, $F(x)$ is the set of all possible outcomes.

3.2.1.2 Observation Platforms

In the final experiments, which are run with partial information, observation platforms are used. These platforms are similar to the air package, but only interact with other objects through detection. Detection provides perfect information regarding specific objects in the JAO environment. We typically consider a global information model in which information is maintained independent of individual battlespace objects. However, the current implementation allows for a more general representation that accounts for distributed information and communication in which information models similar to the global information model are associated with each battlespace object.

3.2.1.3 Targets

Targets are also represented using the hybrid dynamical representation in introduction to Section 3.2. In general, targets have the following discrete state: $q \in \{Known, Unknown, Dead\}$, and may be initialized as known or unknown. The continuous state, for our purposes, is a fixed location, $X = \{x_{loc}\}$ for all q. Figure 3 illustrates the autonomous and controlled jumps associated with targets.

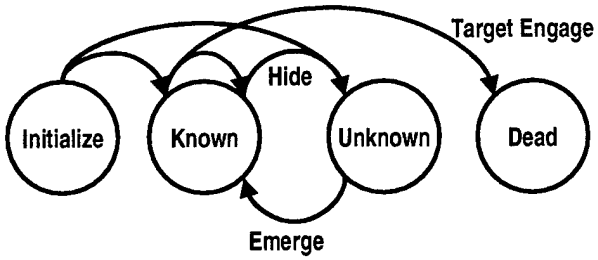


Figure 3 Autonomous and Controlled Jumps Associated with Targets

Autonomous jumps (blue) are only used to represent interactions within a composite model namely engagements with air packages. The controlled jumps (black) are characterized by the controlled jump sets, C_q, and the controlled jump transition maps, F_q. These controlled transitions are governed by a Poisson process, which may be considered a stochastic model of the adversary, allowing the targets to autonomously hide or emerge.

3.2.1.4 Threats

Using the hybrid dynamical representation, threats my be in one of five discrete states, $q \in \{Active, Inactive, Unknown, Repairing, Dead\}$, and may be initialized as active, inactive, or unknown. As with targets, the continuous state is a fixed location, $X = \{x_{loc}\}$ for all q. Figure 4 illustrates the autonomous and controlled jumps associated with threats.

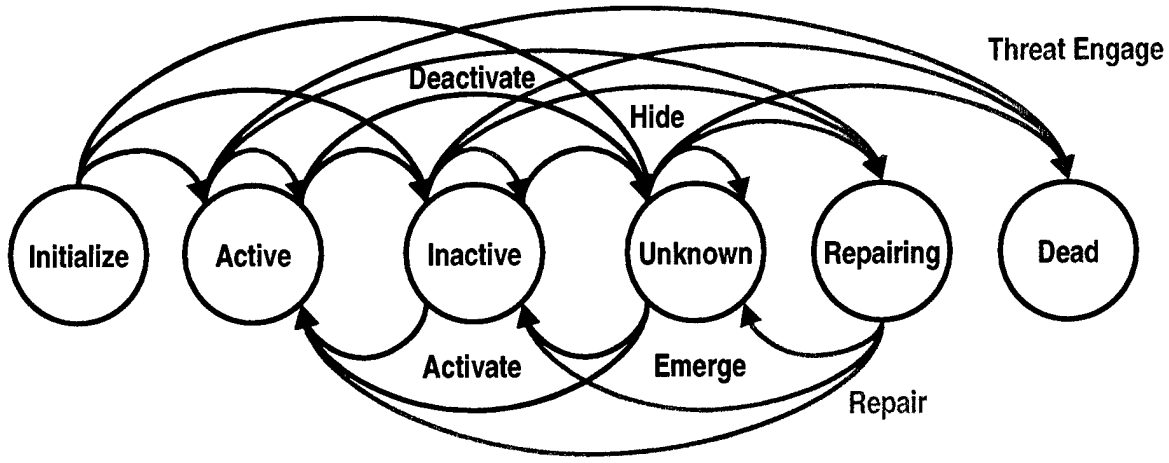


Figure 4 Autonomous and Controlled Jumps Associated with Threats

Autonomous jumps (red) are only used to represent interactions within a composite model namely engagements with air packages. The controlled jumps (black and green) are characterized by the controlled jump sets, C_q, and the controlled jump transition maps, F_q. These transitions are governed by a Poisson process, which may be considered a stochastic model of the adversary, allowing the threats to autonomously active, deactivate, hide, emerge, and repair. In addition, we consider the attack, i.e. SAM launch, transient state during engagement with an air package. We now characterize the autonomous behavior of a threat.

The state transitions that can occur between engagements correspond to *Emerge*, *Hide*, *Activate*, *Deactivate*, and *Repair*. Note that some transitions are only possible during an engagement (e.g., transition to *REAPIRABLE* or *DEAD* from any other state) while other transition are only possible between engagements (e.g., from *REPAIRABLE*). The autonomous transition dynamics are modeled as a stochastic timed automaton, equivalently a continuous time Markov model. Accordingly, a matrix that specifies the rate at which state transitions occur is defined. These rates depend on the time spent in each state and the transition probability, and are represented concisely in the following matrix:

$$
\begin{array}{c}
 \\
\textit{Active} \\
\textit{Inactive} \\
\textit{Repairable} \\
\textit{Unknown} \\
\textit{Dead}
\end{array}
\begin{bmatrix}
\textit{Active} & \textit{Inactive} & \textit{Repairable} & \textit{Unknown} & \textit{Dead} \\
\bullet & \mu_{Inactive|Active} & 0 & \mu_{Unknown|Active} & 0 \\
\mu_{Active|Inactive} & \bullet & 0 & \mu_{Unknown|Inactive} & 0 \\
\mu_{Active|Repairable} & \mu_{Inactive|Repairable} & \bullet & \mu_{Unknown|Repairable} & 0 \\
\mu_{Active|Unknown} & 0 & 0 & \bullet & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

where $\bullet$ is defined so that the rows sum to 0. $\mu_{Inactive|Active}$ is the rate at which an active threat deactivates (i.e., observable but not radiating). $\mu_{Unknown|Active}$ is the rate at which an active threat hides (i.e., becomes unobservable). $\mu_{Active|Inactive}$ is the rate at which an inactive threat activates (i.e., starts radiating). $\mu_{Uknonw|Inactive}$ is the rate at which an inactive threat hides. $\mu_{Active|Unknown}$ is the rate at which an unknown threat activates. The transition rate from *REPAIRED* are functions of the repair rate $\gamma_{REPAIRED}$:

$$\mu_{Active|Repairable} = P_{ACTIVE|REPAIRED} \cdot \gamma_{REPAIR}$$

$$\mu_{Inactive|Repairable} = P_{INACTIVE|REPAIRED} \cdot \gamma_{REPAIR}$$

$$\mu_{Unknown|Repairable} = P_{UNKNOWN|REPAIRED} \cdot \gamma_{REPAIR}$$

where the repair rate is weighted by the probability that the threat will immediately transition into the associated state. These probabilities sum to one

$$P_{ACTIVE|REPAIRED} + P_{INACTIVE|REPAIRED} + P_{UNKNOWN|REPAIRED} = 1$$

indicating that a repaired threat will become *ACTIVE*, *INACTIVE*, or *UNKNOWN*. The transition probabilities are derived from the rate matrix given the time interval Δt_{Inter} using the following matrix equation.

$$P_{Inter}(\Delta t_{Inter}) = e^{Q \cdot \Delta t_{Inter}}$$

where Q is the rate matrix and Δt_{Inter} is the time interval. For a given initial probability distribution over the state of the threat, denoted in vector form by σ_0^T, the distribution after Δt_{Inter}, denoted σ_F^T, is given by

$$\sigma_F^T = \sigma_0^T \cdot P_{Inter}(\Delta t_{Inter})$$

where Δt_{Inter} is the time interval between σ_0^T and σ_F^T. For large enough Δt_{Inter}, σ_F^T will achieve a steady state distribution, which is useful to characterize initial information regarding the threat (i.e., has not been observed or interacted with in a long time).

In the following, we derive the steady state distribution for a threat with the rate matrix Q discussed above. The continuous Markov model is given by

$$\dot{\sigma}^T = \sigma^T \cdot Q$$

with the constraint that σ^T is a proper distribution and the elements sum to one

$$\sum_i \sigma_i = 1$$

Steady state is achieved when $\dot{\sigma}^T = \mathbf{0}$. To compute the steady state probabilities, we define $b^T = [0 \quad 0 \quad 0 \quad 0 \quad 0 \mid 1]$ and $\tilde{Q} = [Q \quad \mathbf{1}]$. The steady state distribution is then given by

$$\bar{\sigma} = \left(\tilde{Q} \cdot \tilde{Q}^T\right)^{-1} \tilde{Q} \cdot b$$

3.2.2 Battlespace Dynamics

When the individual battlespace objects are brought together into a setting where interactions are possible, we form a composite hybrid dynamical model with the same form as the individual objects. However, the composite JAO model subsumes the individual dynamical models and introduces interaction dynamics that depend on two or more objects within the JAO environment. For our purpose, interactions involving more than two participants are decomposed into a sequence of pairwise interactions.

Consider the composite hybrid dynamical model

$$H_c = (Q_c, \Sigma_c, A_c, G_c, V_c)$$

The discrete state space of the composite model is defined as the composition of individual discrete state spaces $Q_c = \prod_i Q_i$, where the subscript i denotes the ith battlespace object. Each state (or mode) of the composite model $q_c \in Q_c$ is associated with a composite dynamical system, $\Sigma_{q_c} \in \Sigma_c = \prod_i \Sigma_i$, a composite autonomous jump set $A_{q_c} \in A_c = \prod_i A_i$, and a controlled jump set $C_{q_c} \in C_c = \prod_i C_i$. For our purposes, the composite dynamical system Σ_{q_c} is a non-interacting parallel composition with the exception of $X_{q_c} \in X_c = \prod_i X_i$, which is the

composite state space on which the autonomous jumps sets $A_{q_c} \subset X_{q_c}$ are defined. Therefore, if the composite state of the dynamical system enters $A_{q_c} \subset X_{q_c}$ (an interaction), the system jumps autonomously to some new mode $p_c \neq q_c$ according to a control mapping G_{q_c}, which is parameterized by the control set V_q. These interactions effect one or more of the battlespace objects.

This hierarchical paradigm simplifies the modeling process by relating directly to the physical reality (i.e., objects and interactions) of the JAO environment. In the following sections, we discuss the interactions between air packages and threats, denoted threat engagement, and interactions air packages and targets, denoted target engagements.

3.2.2.1 Detection Interaction

A detection interaction occurs when an observation platform i is within sensor range of an observable object j

$$\|x_i - x_j\|_2 \leq R_i^2$$

where $x_i \in X_{q_c}$ is the location of an observation platform, $x_j \in X_{q_c}$ is the location of an observable object, and R_i is the sensor range associated with the observation platform i. The result of a detection interaction is a discrete change in the available information regarding the battlespace, but does not otherwise effect the objects. We assume perfect observations.

3.2.2.2 Target Engagement

A target engagement occurs when an air package i arrives at the location of target j

$$\|x_i - x_j\|_2 = 0$$

where $x_i \in X_{q_c}$ is the location of an air package and $x_j \in X_{q_c}$ is the location of a target. The result of a target engagement may only change the discrete state of the target, and does change continuous state of either object, i.e., $X_{q_c} = X_{p_c}$ if q_c is the discrete state prior to the engagement and p_c the discrete state after the engagement. The control set V_{q_c} controls the transition based on the attributes of the interacting objects (i.e., aim points and salvo) and the random component that determines the outcome.

The current model of the target engagement was proposed as part of the enterprise modeling effort. This is a static, analytic model that provides the probability of destroying the

target as a function of the number of Strike aircraft in the air package, the number of aimpoint associted with the target and the strike slavo (i.e., the number of munitions). The probability that air package *i* destroys target *j* is given by:

$$P_{ij} = \left(1 - \exp\left(-numStrike_i / aimPoints_j\right)\right) \cdot \left(1 - \left(1 - P_D^{\text{Target}_j}\right)^{strikeSalvo}\right)$$

where $aimPoints_j$ is an attribute of the Target representing the number of points that need to be hit on a particular target, thus influencing the effectiveness of the air package. P_D^{Target} is the effectiveness of the aircraft's munition. *strikeSalvo* is number of munitions launched by the strike aircraft at the target. This simple static model is sufficient under for the current modeling assumptions (i.e., targets do not defend themselves).

3.2.2.3 Threat Engagement

A threat engagement occurs when an air package *i* is within range of threat *j*

$$\left\| x_i - x_j \right\|_2 \leq R_j^2$$

where $x_i \in X_{q_c}$ is the location of an air package, $x_j \in X_{q_c}$ is the location of a threat, and R_j is the range associated with threat *j*. The result of a threat engagement may change the discrete state of either or both participating objects, but does not change their continuous state, i.e., $X_{q_c} = X_{p_c}$ if q_c is the discrete state prior to the engagement and p_c the discrete state after the engagement. The control set V_{q_c} controls the transition based on the duration of the engagement and the random component that determines the outcome.

The threat engagement is broken into three parts, pre-engagement, engagement, and post-engagement. Pre-engagement transitions account for emergence or activation of a threat with the intent of attacking. Engagement transitions account for the interactive dynamics of an air package with an active and attacking threat. Post-engagement transitions account for hiding or shoot-and-scoot behaviors. Finally, we combine these segments in a single set of transition dynamics and adopt a concise matrix representation.

Pre-engagement transition accounts for emergence with the intent of attacking. The probability that a threat will attack is given by the probability of attacking from any given state

(except *DEAD* and *REPAIRABLE*) times the probability of being in that state. Note, in order to attack, a threat in the *REPAIRABLE* state must have transitioned into another state prior to engagement, otherwise there is no guarantee that sufficient time has elapsed for the "repair" to take place. Therefore, the probability of attack is given by

$$P_{ATTACK} = P_{ATTACK|ACTIVE} \cdot P_{ACTIVE} + P_{ATTACK|INACTIVE} \cdot P_{INACTIVE} + P_{ATTACK|UNKNOWN} \cdot P_{UNKNOWN}$$

This formulation assumes that there is some form of sensing available (possibly visual) that cues threat activation for subsequent attack from *INACTIVE* and *UNKNOWN* states. We also assume that an "attacking" threat is identifiable (e.g., by tracking radar, etc.), thus only threats that attacking are engaged. Therefore, if the threat does not attack, there are no pre-engagement transitions.

$$P^{Pre}_{ACTIVE} = \left(1 - P_{ATTACK|ACTIVE}\right) \cdot P_{ACTIVE}$$
$$P^{Pre}_{INACTIVE} = \left(1 - P_{ATTACK|INACTIVE}\right) \cdot P_{INACTIVE}$$
$$P^{Pre}_{REPAIRABLE} = P_{REPAIRABLE}$$
$$P^{Pre}_{UNKNOWN} = \left(1 - P_{ATTACK|UNKNOWN}\right) \cdot P_{UNKNOWN}$$
$$P^{Pre}_{DEAD} = P_{DEAD}$$

The engagement transition accounts for uncertainty associated with the interaction. In general, engagements may be considered *suppression* or *lethal*. Suppression engagements disable the threat, thus transitioning into the *REPAIRABLE* state while lethal engagements destroy the threat, thus transitioning into the *DEAD* state. We distinguish lethal engagements probabilistically with $P_{DEAD|SUCCESS}$ given a successful attack on the threat. This indicates that sufficient damage was done to transition the threat to *DEAD*. The threat must be *ACTIVE* after the pre-engagement transition to engage, and will transition to either *ACTIVE*, REPAIRABLE, or *DEAD* as a result of the engagement.

$$P^{Engage}_{ACTIVE} = P_{FAIL|ATTACK} \cdot P_{ATTACK} + P^{Pre}_{ACTIVE}$$
$$P^{Engage}_{REPAIRABLE} = \left(1 - P_{DEAD|SUCCESS}\right) \cdot P_{SUCCESS|ATTACK} \cdot P_{ATTACK} + P^{Pre}_{REPAIRABLE}$$
$$P^{Engage}_{DEAD} = P_{DEAD|SUCCESS} \cdot P_{SUCCESS|ATTACK} \cdot P_{ATTACK} + P^{Pre}_{REPAIRABLE}$$

The engagement dynamics are characterized by $P_{FAIL|ATTACK}$ and $P_{SUCCESS|ATTACK} = 1 - P_{FAIL|ATTACK}$, which depend on the interacting air package. Threat that are initially *INACTIVE*, *UNKNOWN*, or *DEAD* do not take part in the engagement.

$$P_{INACTIVE}^{Engage} = P_{INACTIVE}^{Pre}$$
$$P_{UNKNOWN}^{Engage} = P_{UNKNOWN}^{Pre}$$
$$P_{DEAD}^{Engage} = P_{DEAD}^{Pre}$$

Post-engagement transition accounts for deactivating/hiding behaviors after engagement.

$$P_{ACTIVE}^{Post} = P_{ACTIVE}^{Engage} - \left(P_{DEACTIVATE} + P_{HIDE}\right) \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK}$$
$$P_{INACTIVE}^{Post} = P_{DEACTIVATE} \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK} + P_{INACTIVE}^{Engage}$$
$$P_{UNKNOWN}^{Post} = P_{HIDE} \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK} + P_{UNKNOWN}^{Engage}$$

where $P_{DEACTIVATE} + P_{HIDE} \leq 1$.

If the threat is *UNKNOWN*, *REPAIRABLE*, or *DEAD*, there is no post-engagement transition.

$$P_{REPAIRABLE}^{Post} = P_{REPAIRABLE}^{Engage}$$
$$P_{DEAD}^{Post} = P_{DEAD}^{Engage}$$

We combine pre-engagement, engagement, and post-engagement transition probabilities into a concise matrix representation. First, the transition probabilities are aggregated.

$$P'_{ACTIVE} = \left(1 - P_{DEACTIVATE} - P_{HIDE}\right) \cdot P_{FAIL|ATTACK} \cdot \begin{pmatrix} P_{ATTACK|ACTIVE} \cdot P_{ACTIVE} + \\ P_{ATTACK|INACTIVE} \cdot P_{INACTIVE} + \\ P_{ATTACK|UNKNOWN} \cdot P_{UNKNOWN} \end{pmatrix} + \left(1 - P_{ATTACK|ACTIVE}\right) \cdot P_{ACTIVE}$$

$$P'_{INACTIVE} = P_{DEACTIVATE} \cdot P_{FAIL|ATTACK} \cdot \begin{pmatrix} P_{ATTACK|ACTIVE} \cdot P_{ACTIVE} + \\ P_{ATTACK|INACTIVE} \cdot P_{INACTIVE} + \\ P_{ATTACK|UNKNOWN} \cdot P_{UNKNOWN} \end{pmatrix} + \left(1 - P_{ATTACK|INACTIVE}\right) \cdot P_{INACTIVE}$$

$$P'_{REPAIRABLE} = \left(1 - P_{DEAD|SUCCESS}\right) \cdot P_{SUCCESS|ATTACK} \cdot \begin{pmatrix} P_{ATTACK|ACTIVE} \cdot P_{ACTIVE} + \\ P_{ATTACK|INACTIVE} \cdot P_{INACTIVE} + \\ P_{ATTACK|UNKNOWN} \cdot P_{UNKNOWN} \end{pmatrix} + P_{REPAIRABLE}$$

$$P'_{UNKNOWN} = P_{HIDE} \cdot P_{FAIL|ATTACK} \cdot \begin{pmatrix} P_{ATTACK|ACTIVE} \cdot P_{ACTIVE} + \\ P_{ATTACK|INACTIVE} \cdot P_{INACTIVE} + \\ P_{ATTACK|UNKNOWN} \cdot P_{UNKNOWN} \end{pmatrix} + \left(1 - P_{ATTACK|UNKNOWN}\right) \cdot P_{UNKNOWN}$$

$$P'_{DEAD} = P_{DEAD|SUCCESS} \cdot P_{SUCCESS|ATTACK} \cdot \begin{pmatrix} P_{ATTACK|ACTIVE} \cdot P_{ACTIVE} + \\ P_{ATTACK|INACTIVE} \cdot P_{INACTIVE} + \\ P_{ATTACK|UNKNOWN} \cdot P_{UNKNOWN} \end{pmatrix} + P_{DEAD}$$

This set of equations may be concisely represented in matrix form as follows

$$\begin{bmatrix} P'_{ACTIVE} \\ P'_{INACTIVE} \\ P'_{REPAIRABLE} \\ P'_{UNKNOWN} \\ P'_{DEAD} \end{bmatrix}^T = \begin{bmatrix} P_{ACTIVE} \\ P_{INACTIVE} \\ P_{REPAIRABLE} \\ P_{UNKNOWN} \\ P_{DEAD} \end{bmatrix}^T \cdot \begin{bmatrix} P_{Active|Active} & P_{Inactive|Active} & P_{Repairable|Active} & P_{Unknown|Active} & P_{Dead|Active} \\ P_{Active|Inactive} & P_{Inactive|Inactive} & P_{Repairable|Inactive} & P_{Unknown|Inactive} & P_{Dead|Inactive} \\ 0 & 0 & P_{Repairable|Repairable} & 0 & 0 \\ P_{Active|Unknown} & P_{Inactive|Unknown} & P_{Repairable|Unknown} & P_{Unknown|Unknown} & P_{Dead|Unknown} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where the individual terms are defined as follows:

$$P_{Active|Active} = \left(1 - P_{DEACTIVATE} - P_{HIDE}\right) \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK|ACTIVE} + \left(1 - P_{ATTACK|ACTIVE}\right)$$

$$P_{Inactive|Active} = P_{DEACTIVATE} \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK|ACTIVE}$$

$$P_{Repairable|Active} = \left(1 - P_{DEAD|SUCCESS}\right) \cdot P_{SUCCESS|ATTACK} \cdot P_{ATTACK|ACTIVE}$$

$$P_{Unknown|Active} = P_{HIDE} \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK|ACTIVE}$$

$$P_{Dead|Active} = P_{DEAD|SUCCESS} \cdot P_{SUCCESS|ATTACK} \cdot P_{ATTACK|ACTIVE}$$

$$P_{Active|Inactive} = \left(1 - P_{DEACTIVATE} - P_{HIDE}\right) \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK|INACTIVE}$$

$$P_{Inactive|Inactive} = P_{DEACTIVATE} \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK|INACTIVE} + \left(1 - P_{ATTACK|INACTIVE}\right)$$

$$P_{Unknown|Inactive} = P_{HIDE} \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK|INACTIVE}$$

$$P_{Repairable|Inactive} = \left(1 - P_{DEAD|SUCCESS}\right) \cdot P_{SUCCESS|ATTACK} \cdot P_{ATTACK|INACTIVE}$$

$$P_{Dead|Inactive} = P_{DEAD|SUCCESS} \cdot P_{SUCCESS|ATTACK} \cdot P_{ATTACK|INACTIVE}$$

$$P_{Repairable|Repairable} = 1$$

$$P_{Active|Unknown} = \left(1 - P_{DEACTIVATE} - P_{HIDE}\right) \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK|UNKNOWN}$$

$$P_{Inactive|Unknown} = P_{DEACTIVATE} \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK|UNKNOWN}$$

$$P_{Repairable|Unknown} = \left(1 - P_{DEAD|SUCCESS}\right) \cdot P_{SUCCESS|ATTACK} \cdot P_{ATTACK|UNKNOWN}$$

$$P_{Unknown|Unknown} = P_{HIDE} \cdot P_{FAIL|ATTACK} \cdot P_{ATTACK|UNKNOWN} + \left(1 - P_{ATTACK|UNKNOWN}\right)$$

$$P_{Dead|Unknown} = P_{DEAD|SUCCESS} \cdot P_{SUCCESS|ATTACK} \cdot P_{ATTACK|UNKNOWN}$$

Alternately, $P_{Active|Active}$, $P_{Inactive|Inactive}$, $P_{Repairable|Repairable}$, and $P_{Unknown|Unknown}$ may be defined based on the other terms such that each row sums to 1. The parameters of this model are as follows:

- $P_{DEACTIVATE}$ is the probability that an "attacking" threat will deactivate after the engagement;

- P_{HIDE} is the probability that an "attacking" threat will hide (become unknown) after the engagement;
- $P_{FAIL|ATTACK}\left(\pi, \Delta t_{Engage}\right)$ is the probability that the threat survived (i.e., suppression failed) the engagement and is a function of the engagement time, Δt_{Engage}, and the probabilistic state of the associated air package π (Discussed further in following paragraphs);
- $P_{SUCCESS|ATTACK}\left(\pi, \Delta t_{Engage}\right) = 1 - P_{FAIL|ATTACK}\left(\pi, \Delta t_{Engage}\right)$;
- $P_{DEAD|SUCCESS}$ is the probability that a successful threat engagement will destroy the threat;
- $P_{ATTACK|ACTIVE}$ is the probability that an active threat will attack;
- $P_{ATTACK|INACTIVE}$ is the probability that an inactive threat will attack; and
- $P_{ATTACK|UNKNOWN}$ is the probability that an unknown threat will attack.

Consider $P_{FAIL|ATTACK}\left(\pi, \Delta t_{Engage}\right)$, which depends on the state of the associated air package π and the engagement time Δt_{Engage}. This represents the actual engagement dynamics and is based on an aggregate model of the shoot-look-shoot behavior of both the air package and the threat. In this model, the uncertainty associated with the threat engagement depends on the composition of the air package, the state of the threat (as a result of previous missions), and the duration of the engagement (determined from the geometry). The simplified geometry of this interaction is shown in Figure 5. The circle represents the footprint of the SAM's lethal range. Any route through the lethal range of the threat will result in an exposure time that indicates the duration of the engagement, ΔT. The state prior to the engagement is denoted π_0, and the final state is denoted π_F. The engagement dynamics are defined within a composite model of the interacting air package and threat.

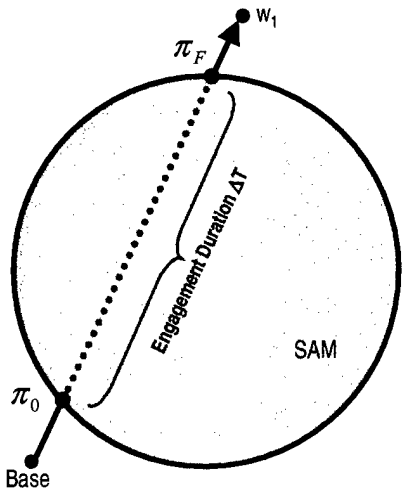


Figure 5 SAM Engagement Model Geometry.

The underlying dynamics of the threat engagement are represented by a set of event (e.g., SAM fires, Weasel fires) that are related by a continuous-time Markov chain, which may be solved analytically using

$$\pi_F = \pi_0 e^{Q\Delta T}$$

where Q is a transition rate matrix describing the engagement dynamics. Q is constructed from a set of parameters that describe the effectiveness of the SAM against each of the aircraft in the package, and the effectiveness of the weasel aircraft against the SAM. The following table summarizes these parameters

Parameter	Symbol	Notional Value
Weasel firing rate	γ_{weasel}	1 shot every 6 min (shoot-look-shoot)
SAM firing rate	γ_{SAM}	1 shot every 6 min (shoot-look-shoot)
Weasel effectiveness	P_D^{SAM}	80%
SAM effectiveness	P_D^{Strike}	90%
SAM effectiveness	P_D^{Strike}	70%

In this formulation, the firing rates account for acquisition, tracking, munition flyout, and reload/turn. The effectiveness accounts for initial detection and munition effectiveness. This model may be tuned and is easily extended; however, the desirable behavior has been observed

with only notional parameter values. Given these parameters, the sparse Q transition rate matrix is represented as follows:

$$
\begin{bmatrix}
a_{11} & 0 & & \gamma_{SAM}P_D^{Weasel} & & & \gamma_{Weasel}P_D^{SAM} & & & & & \\
 & a_{22} & 0 & & \gamma_{SAM}P_D^{Weasel} & & & \gamma_{Weasel}P_D^{SAM} & & & & \\
 & & a_{33} & & & \gamma_{SAM}P_D^{Weasel} & & & \gamma_{Weasel}P_D^{SAM} & & & \\
 & & & a_{44} & \gamma_{SAM}P_D^{Strike} & & & & & 0 & & \\
 & & & & a_{55} & \gamma_{SAM}P_D^{Strike} & & & & & 0 & \\
 & & & & & 0 & & & & & & 0 \\
 & & & & & & 0 & & & & & \\
 & & & & & & & 0 & & & & \\
 & & & & & & & & 0 & & & \\
 & & & & & & & & & 0 & & \\
 & & & & & & & & & & 0 & \\
 & & & & & & & & & & & 0
\end{bmatrix}
$$

3.3 HYBRID DYNAMICAL MODEL IMPLEMENTATION

Having defined the hybrid dynamical framework, individual asset dynamics, and the parallel composition of these dynamical entities through interaction dynamics, the plant dynamics were implemented in ALPHATECH's BMC^3 Development Environment, whose graphical user interface is illustrated in Figure 6 below. This is a discrete event simulation environment, which is written in the JAVA programming language. JAVA provides seamless portability across multiple hardware and software platforms, ease in process distribution, and programming efficiency due to the language's inherent object oriented structure.

The plant's object oriented design provides a flexible environment for adding and modifying simulation objects and events. The simulation objects contain individual information and interact as separate entities. Likewise, the events encapsulate specific behaviors and dynamics in the simulation, providing their own execution logic. This makes for a flexible system which may be easily extended or modified without changing the existing simulation framework.

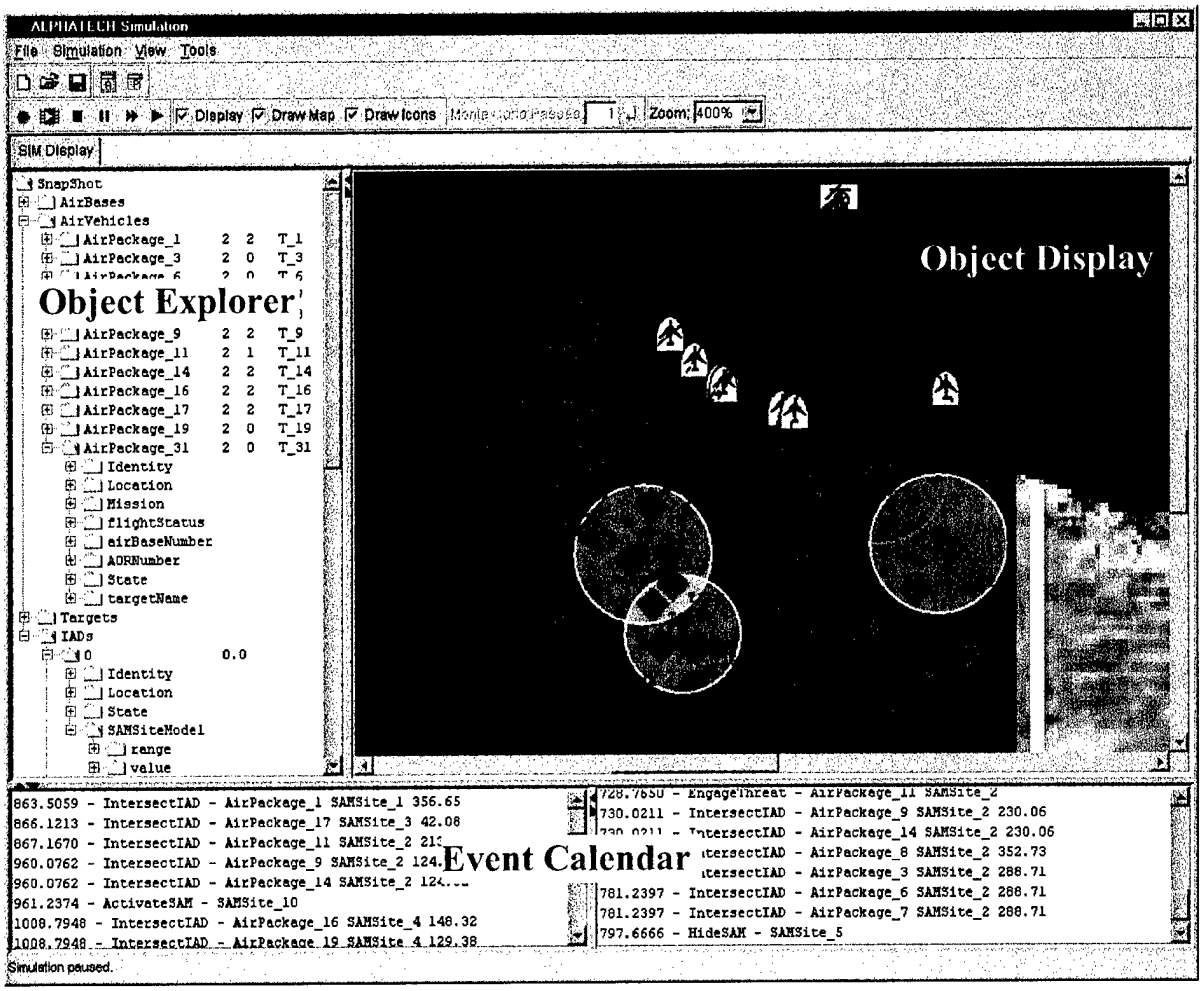


Figure 6 ALPHATECH's BMC^3 Development Environment GUI

3.3.1 Discrete Event Simulation

ALPHATECH's BMC^3 Development Environment is based on the discrete event architecture illustrated in Figure 7 [CassLaf99]. At the heart of the simulation is the event calendar, a time-sorted list of events waiting to be executed. The simulation is driven ahead in time by continually removing the next event from the calendar, updating the simulation time to that of the event by updating any continuous variables (such as position of air packages), and finally processing the event. The event is created with the knowledge of which simulation objects are involved in it, and any other information that may dynamically affect its behavior. Therefore, when the simulator instructs it to do so, the event can execute itself according to its inherent dynamics, using the current state of the simulation. Depending on the specific event's execution, it may update the state of the simulation by changing the state of any relevant simulation objects, adding new objects to the simulation, removing objects, or by adding new events to the calendar. This process continues until the event calendar is empty or any user defined stop conditions are met.

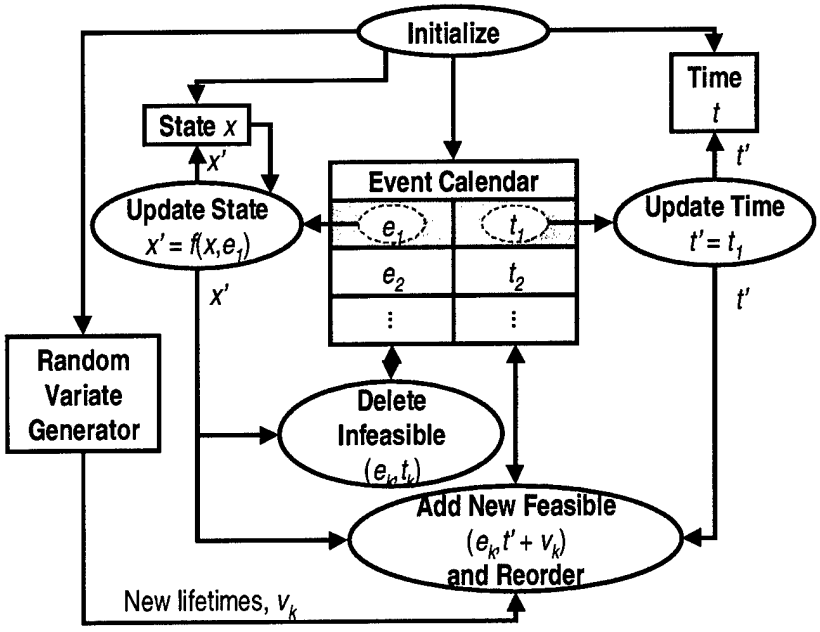


Figure 7 Discrete Event Simulation Framework
(copied form Reference[CassLaf99])

3.3.2 Plant/Controller Interface

When an event is executed that requires guidance from the controller, the plant accomplishes this via an interface to the controller. This interface between the plant and the controller maintains the integrity of each as separate software entities. When a control decision is needed, it is the plant's responsibility to collect the current known or estimated state of all simulation objects into a data structure required as input to the controller, which is sent via the

interface. Depending on arguments specified by the user, the information passed to the controller may be collected by the plant in two different ways. Most experiments performed assume perfect state information about all objects in the simulation. In this case, the plant extracts the true state of all objects from the simulation to create the data structure sent to the controller. When modeling information collection, however, a data structure is maintained with the current estimated or observed state of all simulation objects, which gets updated at the appropriate times by all surveillance aircraft. In this case, a new deep copy of this data structure is made by the plant and passed via the interface to the controller. It is then the controller's responsibility to return its guidance to the plant via the interface, using the same type of data structure it was passed. However, since the controller only has the ability to affect the actions of air packages, only that type of object is returned to the plant. If the controller decides to configure and task new air packages, objects are returned to instruct the plant how to do so. When it needs to modify the missions of existing air packages, any changes will be reflected in the corresponding air package objects' missions returned as guidance. The plant must then incorporate those changes into the real air packages. The controller does not have the ability to modify or create any of the simulation objects itself, but rather instructs the plant how to do so.

As described above, the plant and controller have a well defined client/server relationship. The plant's simulation runs as a client, which calls the controller as a server when it needs guidance. Each is a separate software entity, although they do not have to be run as separate processes. In fact, the plant and controller almost always run in the same process when performing experiments to save the overhead of communicating over a network. The interface described above makes it easy to maintain this relationship, as represents the network between the client and server.

3.3.3 Distributed Processing

Some of the controller algorithms implemented require a great deal of computation time. No matter how streamlined and efficiently the code is written, it is just infeasible to use only one process to get some control decisions due to the computational requirements. This becomes even more of a problem as the size of the problem being solved grows. One way of dealing with these issues of computation time and problem scalability is to distribute the controller's processing

duties among multiple CPUs, thereby decreasing the overall time it takes to close the loop. This can be done in a variety of ways, depending on how a specific controller works.

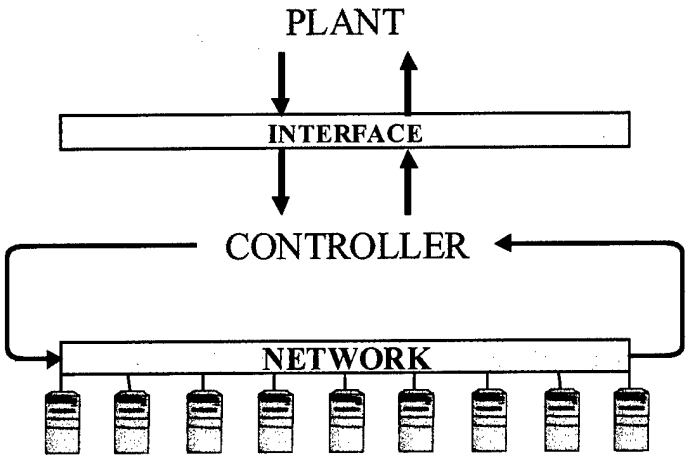


Figure 8 Distribution of Controller Processing Across Multiple Platforms

One example of a controller that benefits from distributing its processing is the Maximum Marginal Return (MMR) algorithm (see Section 4.3.1). Before allocating each aircraft, the MMR must evaluate the outcome of adding it to every air package in the current mission queue to determine how the asset can best be used. Rather than calling its predictor object to estimate the performance expectations one option (asset to air package assignment) at a time, the MMR can evaluate each option in parallel by distributing its predictor calls over multiple processors. In this case, if twenty options needed to be evaluated and ten machines were available, distributing the processing should be ten times faster than evaluating all twenty options in a single process.

4 JAO CONTROLLER FORMULATION

In this chapter, the JAO controller formulation details will be presented. The discussion begins with a general discussion of the control framework used for this research. Then, some proof of concept experimental results that guided our controller research framework will be presented. Having established the control and research framework, the details of the controller formulations and implementations will be presented. At the end of this chapter, a scalability analysis will be presented to assess whether near real-time computation performance is achieved.

4.1 CONTROL FRAMEWORK

The JAO environment is a uncertain dynamical system that has the following attributes: control decisions made over time; probabilistic transition from one state to the next, which is dependent on the choice of control; and rewards/costs that are accumulated during each transition, which is dependent on control and state transition outcome. Thus, the tasking of air packages in a JAO environment can be viewed as a sequential decision problem where each decision is based on the observations of certain discrete events.

This class of problems can be formulated as a Markov decision problem [B96]. The principal approach for solving such problems is Stochastic Dynamic Programming (SDP). Using the SDP formulation, an optimal control solution is computed off-line, and on-line computation is reduced to feedback rule evaluation or table lookup interpolation. However, it is well known that this approach suffers from the *curse of dimensionality* and is intractable for realistic sized JAO problems.

A subtle but significant attribute of the SDP formulation is that it produces control strategies that anticipate the effects of future contingencies, and evaluates the possible actions at all future states. The algorithm accomplishes this by modeling the future information arrival and control decisions. It is this fact that produces *proactive* versus *reactive* control behaviors. This proactive attribute is imperative for stable and agile control of the JAO enterprise because future information arrival and control opportunities are dependent on stringent spatial, temporal, and coordination constraints.

Given the strengths and weakness of the SDP formulation, there has been a great deal of research on Approximate Dynamic Programming (ADP) methods in recent years. These methods generally maintain the SDP structure, but use a variety of techniques to approximate the optimal cost-to-go. Accordingly, ADP algorithms have been applied to a variety of dynamic

decision problems [B99], [BTW97], [Patek99], [BC98], [BC99]. The goal of this research is to extend these ADP algorithms to the JAO context. In the following paragraphs, the general SDP and ADP formulations will be presented.

Consider a discrete-time version of a dynamic decision problem,

$$x_{k+1} = f_k(x_k, u_k, w_k)$$

where x_k is the state taking values in some set X_k, u_k is the control to be selected from a finite set $U_k(x_k)$, w_k is a random disturbance, and f_k is a given function. We assume that the disturbance w_k, $k=0,1,...$ has a given distribution that depends explicitly only on the current state and control. Define a control policy, which is a sequence of feedback functions that map each state x_k to control u_k:

$$\pi_k = \{\mu_k(x_k), \mu_{k+1}(x_{k+1}), ..., \mu_{k+N-1}(x_{k+N-1})\}$$

thus, the control at time k is $u_k = \mu_k(x_k) \in U_k(x_k)$. In the N-stage horizon problems considered herein, the single-stage cost function is denoted by $g_k(x_k, \mu_k(x_k), w_k)$ and the terminal cost function is denoted by $G_{k+N}(x_{k+N})$. The cost-to-go for policy π starting from state x_k at time k can be computed as follows:

$$J_k^{\pi}(x_k) = E\left\{G_{k+N}(x_{k+N}) + \sum_{i=k}^{k+N-1} g_i(x_i, \mu_i(x_i), w_i)\right\}$$

and can be represented in the SDP recursion format as follows

$$J_k^{\pi}(x_k) = E\{g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}^{\pi}(f_k(x_k, \mu_k(x_k), w_k))\}$$

for all k and with the initial condition

$$J_{k+N}^{\pi}(x_{k+N}) = E\{G_{k+N}(x_{k+N})\}$$

The N-stage, SDP solution is as follows

$$\pi_k^* = \arg \min_{\pi_k, u_k \in U_i(x_i)} E\{g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}^{\pi}(f_k(x_k, \mu_k(x_k), w_k))\}$$

The computational feasibility of the SDP recursion depends on the number of future state realizations required to describe the system. Figure 9 provides a graphic illustration of the SDP recursion and tree structure of possible future state realizations. To illustrate the number of states required, assume that there are N targets, M air packages, and that we simplify physical position descriptions to describe only the N positions of the targets. Then, the number of possible combinations of positions is M^N, and the number of possible uncollected target sets at a given time is 2^N, resulting in numbers of states $(2M)^N$. For modest numbers of assets and targets, the number of states far exceeds our capability for computing and/or storing the resulting optimal decision rules.

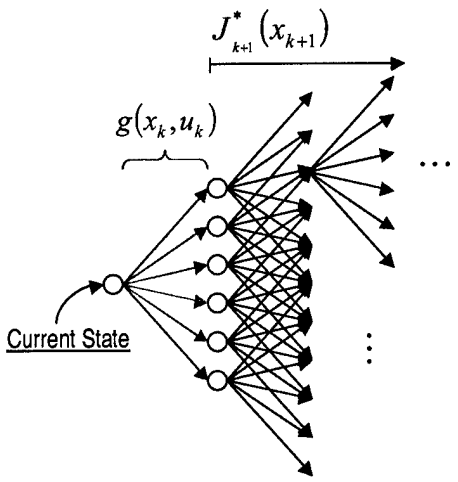


Figure 9 SDP Recursion

In an attempt to overcome the SDP *curse of dimensionality*, the ADP algorithm replaces the control mapping for times $k+1 \rightarrow k+N-1$ with some approximate mapping $\overline{\mu}_i(x_i)$. Additionally, the ADP algorithm is solved forward in time, and is computed at the actual state x_k versus all possible states at time k. As Appendix 8.1 presents, there are a variety of approaches to approximating future control maps. The approach adopted for this research is to generate an approximate control policy that maps a subset of future state realizations to a subset of control actions.

Thus, the ADP algorithm has the following policy:

$$\pi_k^{ADP} = \{u_k(\mathrm{x}_k), \overline{\mu}_{k+1}(\mathrm{x}_{k+1}), \ldots, \overline{\mu}_{k+N-1}(\mathrm{x}_{k+N-1})\}$$

Using this policy, the approximate optimal control solution at time k is

$$u_k^{\mathrm{ADP}} = \arg\min_{u_k \in U_i(x_i)} \mathrm{E}\left\{\mathrm{g}_k(\mathrm{x}_k, \mu_k(x_k), w_k) + \mathrm{J}_{k+1}^{\pi^{ADP}}(\mathrm{f}_k(\mathrm{x}_k, u_k(x_k), w_k))\right\}$$

Thus, the ADP policy is a one step-look-ahead policy with the optimal cost-to-go approximated by the cost-to-go of the base policy. The ADP algorithm computes the best control at the current state x_k at time k by balancing the current cost with an approximate cost-to-go using approximations to model future control decisions. A graphical illustration of the ADP approach is illustrated in Figure 10 where the tree structure of the cost-to-go has been replaced with an approximate cost-to-go.

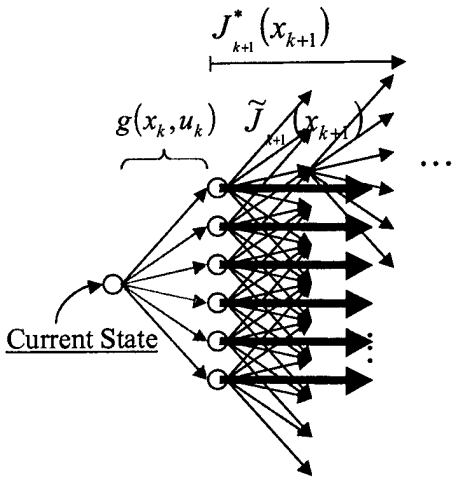


Figure 10 NDP Solution Structure

By approximating the future control maps and by solving the problem forward in time for the actual state x_k, the ADP algorithm significantly reduces the computational complexity of the SDP framework. Again, SDP considers all of the possible states and computes a tentative decision for each possible state, whereas ADP only computes decisions for states that actually occur in the scenario. Thus, the number of states considered by ADP is much smaller; however, the drawback of this approach is the solution must be determined in real-time.

Having defined the ADP framework, the difficulty remains of computing the expectation in the above optimization. Given the complexity of the JAO problem and the fact that the control solution will have to be computed in real-time, only an estimate of the expectation can be computed. Accordingly, the Q-factor is introduced:

$$Q(x_k,u_k)=\mathrm{E}\left\{\mathrm{g_k}(\mathrm{x_k},\mu_k(x_k),w_k)+\mathrm{J}_{\mathrm{k+1}}^{\pi^{ADP}}(\mathrm{f_k}(\mathrm{x_k},u_k(x_k),w_k))\right\}$$

For the reasons stated above, only an estimate for the Q-factor $\hat{\mathrm{Q}}(\mathrm{x_k},\mathrm{u_i})$ is obtained. Thus, given the estimate Q-factor $\hat{\mathrm{Q}}(\mathrm{x_k},\mathrm{u_i})$ corresponding to each candidate control $u_k \in U_k(x_k)$, the ADP control at time k for state x_k is

$$u_{\mathrm{k}}^{\mathrm{ADP}}=\arg\min_{u_k\in U_i(x_i)}\mathrm{E}\{\mathrm{Q}(\mathrm{x_k},u_k(x_k))\}$$

In summary, the ADP algorithm provides the control framework for this research. This technique has been shown to illustrate operationally consistent, proactive behaviors for relevant military applications. The focus of this research is to extend the ADP method to the problem of

JAO, and in doing so, develop ADP algorithms that exhibit optimal behavior in real-time or near real-time for realistic JAO scenarios. As a final note, given the approximation illustrated above, the ADP algorithm is not as ambitious as the SDP, and only provides modest guarantees of near-optimality [BTW97]; in fact, it is an intermediate methodology between Model Predictive Control (MPC) and SDP.

4.2 PROOF OF CONCEPT EXPERIMENTS AND RESEARCH FRAMEWORK

In this section, the proof of concept experiments that guided the development of ADP control techniques for the JAO problem will be presented. Following the proof of concept experiment discussion, the research framework adopted for this program will be presented.

4.2.1 Proof of Concept Experimental Results

At the beginning of this research program, proof of concept experiments were performed for the purpose of identifying key technology gaps in the state-of-the-art of ADP with respect to the JAO domain. By identifying the key technology barriers, the research and development was focused on mitigating these barriers so as to satisfy the research objective of providing military commanders with real-time, near optimal control strategies for air-to-ground operations. In this section, the proof of concept experimental results that guided the development of ADP control techniques for the JAO problem will be presented. Since a majority of these initial experimental results have been documented in conference publications, most of the details are contained in the Appendices and the summary of results and lessons learned will be summarized here.

The approach used to establish the proof of concept experiments was to apply ADP techniques that have been applied to other relevant military problems. As part of AFOSR's New World Vistas (NWV) program, ALPHATECH performed basic research on ADP control techniques for the problem of sensor asset scheduling. Under this program, ADP control techniques were developed that optimize the collection of data by multiple sensor platforms based on requests of multiple end-users. The controller dynamically replans the paths of sensor platforms as the result of dynamic requests for data, failure of individual sensors (thereby providing fault tolerance), and failure of sensors to collect individual pieces of data due to unpredictable obscuration effects such as weather. Many of the results from this program are contained in two DARPA Advances in Enterprise Control papers that appear in Appendices 8.2 and 8.3.

Given this successful application of ADP techniques to military relevant problems, the next logical step was to apply the ADP techniques developed under the AFOSR program to the problem of orchestrating a 24 hour air-to-ground campaign in a risky environment. However, the multi-vehicle scheduling problem does not map one-to-one with the air-to-ground problem because this problem is much larger and contains a richer set of dynamics. For one, this problem requires the formation and tasking of air packages in an environment where risk and reward depend on the air package composition. Furthermore, since air packages pose a risk to enemy assets and enemy assets pose a risk to air packages, considerable coupling exists between battlespace assets. Given the fact that multiple turns of the aircraft will be required to achieve the operational objectives, this coupling remains dominant through the air-to-ground campaign.

For the proof of concept experiments, the rollout algorithm [B99], [BTW97] was chosen for implementation on a reduce-order JAO problem. The rollout algorithm—which has been used for a wide variety of dynamic decision problems [B99], [Patek99], [BC99], [BC98], [BC99] — is a technique that exploits knowledge of a suboptimal decision rule to obtain an approximate cost-to-go for use in ADP framework. The rollout algorithm approximates control mapping for times $k+1 \rightarrow k+N-1$ with a baseline heuristic $\bar{\mu}(x_i)$. Thus, the rollout algorithm computes the best control at the current state x_k at time k by balancing the current cost with an approximate cost-to-go using a baseline heuristic to model future control decisions. To generate the estimate of Q-factor, Monte Carlo evaluations were performed by simulating the base policy in real-time, i.e. simulation-in-the-loop.

The rollout algorithm is applied to a small JAO scenario, Figure 11, that includes limited assets, risk/reward that is dependent on package composition, basic threat avoidance routing, and multiple targets, some of which are fleeting and emerging. Simulation results illustrate the benefits of the approximate optimal control strategy. It is shown that the

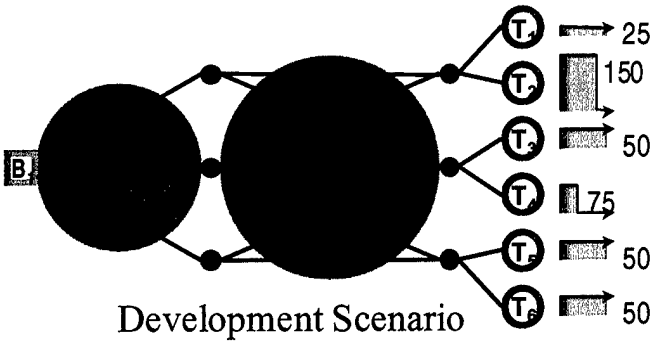


Figure 11 Reduced-Order JAO Scenario Used for Proof of Concept Experimentation

rollout strategy provides statistically significant performance improvements over strategies that do anticipate future information arrival and control decisions. The performance improvements were attributed to the fact that the rollout algorithm is able to learn near-optimal behaviors—establishing combat air patrol over time critical areas, staging packages and opening attack corridors to manage friendly asset attrition, aggressively prosecuting fleeting targets, and reserving assets for contingencies that are not modeled in the baseline heuristic. The details of this experiment and the results obtained are contained in Appendix 8.4.

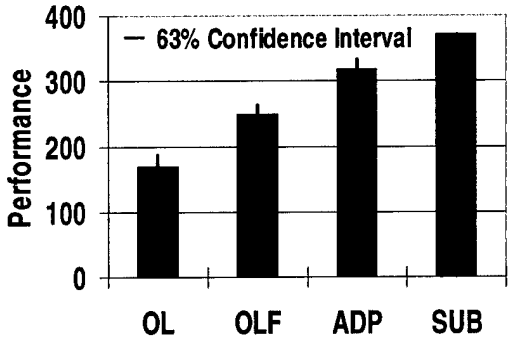


Figure 12 Proof of Concept Performance ADP for Reduced-Order JAO Scenario

Having shown that ADP strategies can produce operationally consistent, proactive control solutions, the question remains whether this current ADP implementation using rollout is feasible for a realistic sized JAO scenario. Figure 13 illustrates the scalability assessment performed on this ADP implementation. From this figure, it is seen that that real-time computation performance is not feasible for larger scenarios. Furthermore, when considering a richer set of dynamics, it is expected that the computation complexity will grow by several orders of magnitude.

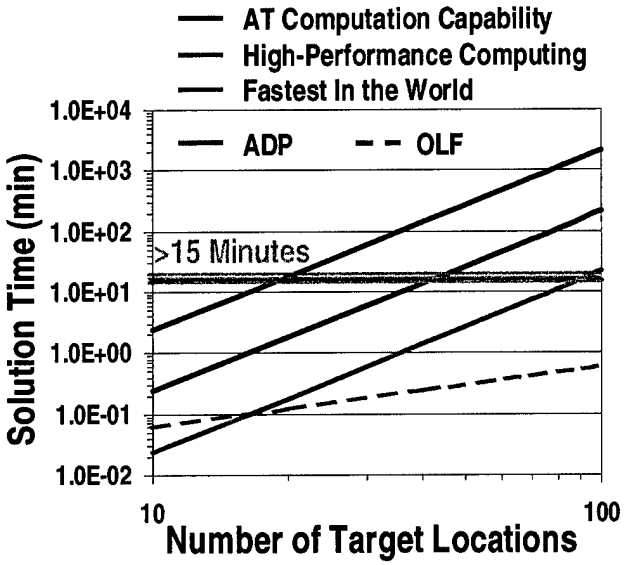


Figure 13 Scalability Assessment of Proof of Concept ADP Implementation

From this proof of concept experiment, the following lessons were learned relative to JAO control:

- **Proactive and Reactive Control** provides near-optimal performance in situations with abundant opportunity and time to react to uncertain future information arrival;
- **Proactive Control** provides near-optimal performance in situations with limited opportunity and/or time to react to uncertain future information arrival:
  - High attrition environment
  - Control response delays, i.e. inertia, ≥ significant event time-scales
  - Information delay
- **Key Proactive JAO Behaviors**: *Positioning assets now for future opportunity*
  - Preparing battlespace
  - Reserving assets
  - Geographically positioning assets
- **Computational Performance:**
  - ADP using combinatorial rollout to search control space and temporal rollout, i.e. simulation-in-the-loop, for prediction is infeasible for 100 DMPI scenario
  - The reactive controller implemented provides rapid replanning, and is feasible for 100 DMPI scenario and has considerable margin

In summary, ADP technique known as rollout, which was developed under the AFOSR NWV program, was applied to a reduced-order JAO scenario that includes limited assets, risk/reward that is dependent on mission composition, basic threat avoidance routing, and multiple targets, some of which are fleeting and emerging. Simulation results illustrate the benefits of the ADP control strategy. It is shown that the proactive ADP strategy provides statistically significant performance improvements over a reactive feedback strategy by developing operationally consistent control strategies that anticipate likely contingencies and position assets for opportunities of recourse. However promising these results are, the current implementation of the rollout algorithm is not computationally feasible for realistic JAO scenarios.

4.2.2 Control Research Framework

The proof of concept experiments in the above section highlighted ADP performance both in terms of behaviors and computation complexity. It was shown that ADP control strategies could produce proactive, operationally consistent behaviors but scalability remains the key technical barrier of using this technique for realistic sized JAO problems. As a result, the

bulk of this research was devoted toward reducing the computational complexity of the ADP approach while maintaining the operationally consistent behaviors. Based on the lessons learned, a two pronged approach for reducing the problem complexity was pursued:

1. Reduce control problem size where appropriate, and
2. Improved the efficiency of the ADP algorithms.

As discussed in the previous section, one of the lessons learned was that both reactive and proactive control provides near-optimal control strategies in situations where there is abundant opportunity and time to react to uncertain future information arrival. However, the reactive control approach can produce this solution in a fraction of the time that it takes the proactive controller approach. Thus, in situations where there is abundant opportunity and time to react to uncertain future information arrival, it makes sense to use a reactive versus proactive control approach. The lessons learned also identified situations in which a proactive control approach is required in order to proved near-optimal control strategies. These situations include environments that exhibit high attrition and significant control response and information delays relative to the battlespace time scale. With this intuition into the JAO control problem, a hybrid, multi-rate control architecture was adopted that tailors the application of control, i.e. reactive or proactive, at a time scale that is appropriate to the battlespace situation at hand.

Figure 14 illustrates the hybrid, multi-rate architecture chosen for this research. To implement this architecture, three types of information must be specified *a priori*. First, the significant events at which control loop closures are to occur must be defined. For each significant event, a set of control algorithms must be defined, and finally, the loop closure rate for each significant event must be defined. In this figure, event-based loop closures are denoted by **E** where temporal-based loop closures are denoted by **T**. By adopting this architecture, the control problem complexity is substantially reduced for situations that do not require advanced ADP algorithms.

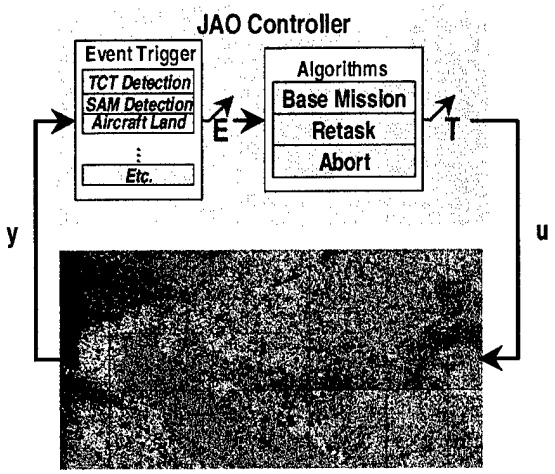


Figure 14 Hybrid, Multi-Rate Control Architecture

The implementation of this control architecture provided an immediate reduction in the control problem complexity for certain situations. However, as noted in the lessons learned, situations with limited opportunity and/or time to react to uncertain future information arrival require proactive control strategies to achieve optimal performance. Thus, for these situations, the complexity reduction must be achieved by developing fast and efficient ADP algorithms that still exhibit the desired operationally consistent behaviors. The approach adopted for developing efficient ADP algorithms was to exploit the natural decomposition, which is illustrated in Figure 15, between the control space search and the performance prediction problems of the ADP framework. Given this natural decomposition, parallel research initiatives that focused on reducing the complexities of these problems were conducted. Figure 16 highlights some of the technologies that were investigated as part of the complexity mitigation.

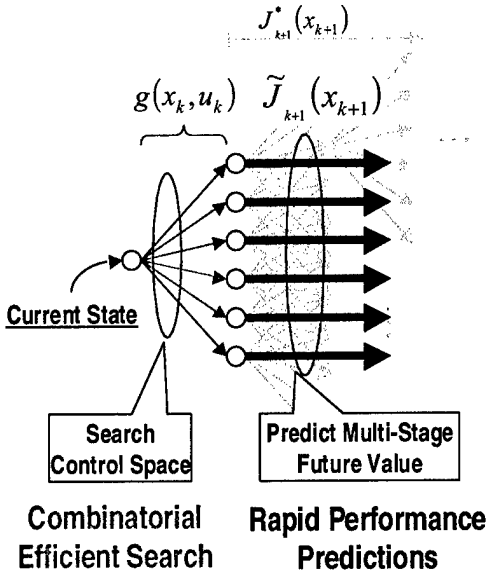


Figure 15 ADP Complexity Mitigation Approach

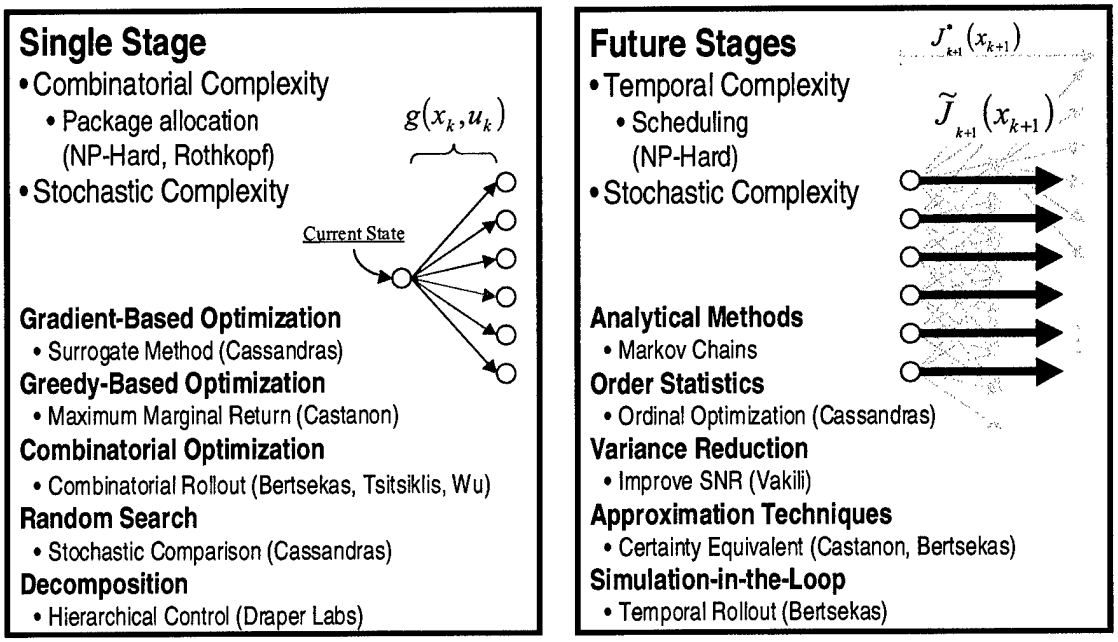


Figure 16 Enabling Technologies Investigated to Mitigate ADP Complexities for JAO Problem

In summary, the proof of concept experiments showed that ADP control strategies could produce proactive, operationally consistent behaviors but scalability remains the key technical barrier of using this technique for realistic sized JAO problems. As a result, the bulk of this research was devoted towards reducing the computational complexity of the ADP approach while maintaining the operationally consistent behaviors. One immediate complexity reduction was achieved by adopting a hybrid, multi-rate control architecture that tailors the application of control, i.e. reactive or proactive, for the battlespace situation at hand; however, additional complexity reduction is required. Thus, the research was focused on developing fast and efficient combinatorial assignment and prediction models that form the foundation of the ADP algorithm. Figure 17 illustrates the control architecture in the context of the ADP decomposition, i.e. assignment and prediction. It is the goal of this research to develop a spectrum of ADP control strategies that can be mixed and matched to provide a broad range of performance and computational complexity. The details of the different combinatorial assignment and analytic prediction models developed under this research follows in the subsequent sections.

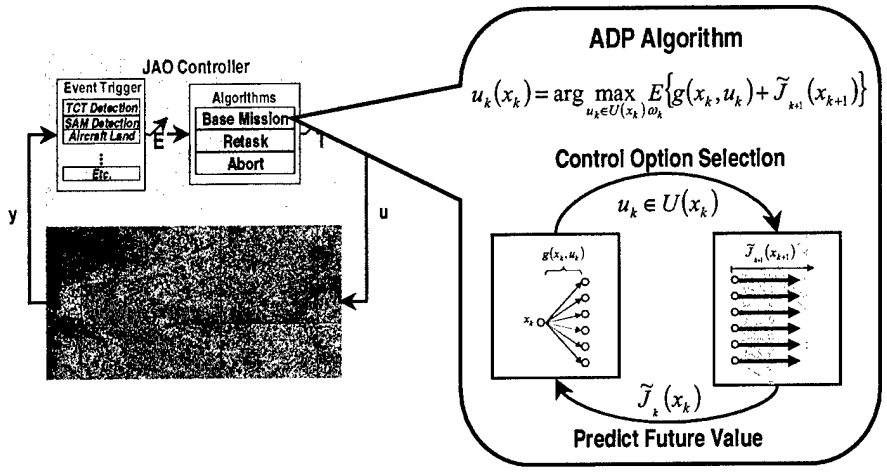


Figure 17 ADP Research Framework in the Context of the Hybrid, Multi-Rate Control Architecture

4.3 COMBINATORIAL OPTIMIZATION ALGORITHMS

In this section, the combinatorial optimization algorithms that were incorporated into our hybrid, multi-rate control architecture are presented. These algorithms were selected based on a trade study analysis of different combinatorial optimization approaches for a 1-stage problem.

Details of this trade study appear in Appendix 8.5. From this trade study, three approaches were chosen: combinatorial rollout, maximum marginal return, and the surrogate method. The formulation of each of these techniques will be presented in the subsequent sections. We first begin with a general discussion of combinatorial optimization.

As noted in Section 4.1, optimization of the Q-factor is fundamental to our control formulation. In a standard SDP implementation, the control space is enumerated, thus guaranteeing an optimal solution. However, in the JAO problem, the complexity of the control space and the requirements for near real-time computation make explicit enumeration infeasible for problems of interest. Therefore, in the course of our proof-of-concept experiments, we investigated several alternatives to direct enumeration. These alternatives are combinatorial optimization techniques, adapted to the structure of the JAO problem, which are approximate optimization techniques. Our rationale for using approximate combinatorial optimization is that the JAO problem, even in the absence of dynamics, is provably NP-Hard, as 3-D matching can be reduced in polynomial time to instances of the JAO problem. We consider three specific approximate combinatorial techniques:

- A greedy algorithm, based on maximum marginal return, using resource by resource decomposition;
- Combinatorial rollout, which is an approximate technique for incrementally building a solution;
- Surrogate method, which embeds the combinatorial optimization in a larger continuous space optimization.

We describe each of these techniques in greater detail below.

To better define the combinatorial problem, consider the problem of optimizing a known function $Q(x_k, u_k)$, over a set of feasible controls $u_k \in U(x_k)$ for a given state x_k. In the JAO case, each control corresponds to a set of resource allocation pairs $(N_i^{Strike}, N_i^{Weasel})$ to possible missions i; that is,

$$u_k = \left\{ \left(N_0^{Strike}, N_0^{Weasel}\right), \left(N_1^{Strike}, N_1^{Weasel}\right), \ldots, \left(N_T^{Strike}, N_T^{Weasel}\right) \right\}$$

in which N_i^{Strike} strike aircraft and N_i^{Weasel} weasel aircraft are assigned to the i-th target for all i. Due to the potential coupling in effectiveness across missions which fly across similar air

defenses, function Q cannot be decomposed as an additive sum of effectiveness across missions. The combinatorial optimization problem becomes

$$\max_{u_k \in U(x_k)} Q(x_k, u_k)$$

subject to resource constraints on the total availability of strike and weasel aircraft, which may be written as

$$\sum_i N_i^{Strike} \le N^{Strike}(x_k)$$

$$\sum_i N_i^{Weasel} \le N^{Weasel}(x_k)$$

where $N^{Strike}(x_k)$ and $N^{Weasel}(x_k)$ correspond to the availability of each aircraft type at a given state x_k.

4.3.1 Maximum Marginal Return

The first algorithm we discuss is the Maximum Marginal Return algorithm (MMR). The basis for this algorithm is the following optimization problem, which is a simplified mathematical model of the JAO problem that provides explicit approximations for the function *Q(x,u)* for a single wave scenario.

Assume that we have present $k = 1,\ldots,K$ SAM sites in the scenario, $t = 1,\ldots,T$ targets, $w = 1,\ldots,W$ weasels, and $s = 1,\ldots,S$ strike aircraft in the scenario. The simplified model assumes that trajectories for each target t are known, and have a sequence of SAMs associated with each trajectory. When a weasel interacts with SAM k while headed for target t, the probability that the SAM is destroyed is given by q_{kt}. We make the probabilistic assumption that interactions between SAMs and weasels are independent events. In this case, given m weasels headed for target t, the probability that SAM k survives is given by:

$$P_s(k) = (1 - q_{kt})^m$$

Similarly, given a full set of missions, with m_t weasels headed for target t, the overall probability that SAM k survives is given by

$$P_s(k) = \prod_t (1 - q_{kt})^{m_t}$$

This simple equation requires the additional assumption that risk to weasels is negligible in these interactions.

The second part of the model represents the interactions between SAMs and strike aircraft, and between strike aircraft and targets. Let p_t denote the probability that a strike aircraft which reaches target t destroys the target. Let r_{kt} denote the probability that if SAM k is still alive, it will destroy a strike aircraft headed for target t. Note the dependence on the target, which represents how strongly a specific SAM can interact with the route headed for target t. Assuming again that interaction events between SAMs and strike aircraft, between strike aircraft and targets, and between weasel aircraft and SAMs are mutually independent, we obtain the following expressions: Let n_t denote the number of strike aircraft allocated to target t. Then, the probability that a strike aircraft headed for target t reaches target t is given by $P(t)$:

$$P(t) = \prod_k (1 - r_{kt} \prod_{t'} (1 - q_{kt'})^{m_{t'}})$$

and the probability that target t survives is given by $P^S(t)$:

$$P^S(t) = (1 - P(t))^{n_t}$$

Given target values V_t the combinatorial optimization problem of interest is:

$$\max_{(m_t, n_t)} \; J(\{(m_t, n_t)\}) = \sum_t V_t (1 - P^S(t))$$
$$\text{subject to} \quad \sum_t n_t \le S, \; \sum_t m_t \le W$$

The above objective function exhibits the coupling between packages headed for different targets affected by common SAMs, plus the coupling between weasels and strike aircraft in packages. Consideration of this objective function reveals that if the weasel allocations to each package are fixed, then the objective function becomes a separable objective function over the strike aircraft content, where each separable component is concave. That is, if we fix m_t, the objective function becomes

$$J(\{(m_t, n_t)\}) = \sum_t J_t(n_t)$$

where

$$J_t(n_t) = V_t(1 - (1 - P^S(t))^{n_t})$$

has a concave envelope. This means that finding the optimal strike aircraft allocation is a simple optimization problem if the weasel allocation is known. This problem is solvable optimally using

a maximum marginal return algorithm, which assigns strike aircraft incrementally to the target that offers the greatest increase in performance per strike aircraft assigned. This approach suggests an alternating procedure, which alternates between selecting weasel aircraft assignments and strike aircraft assignments. Unfortunately, fixing the strike aircraft assignment does not decouple the weasel assignment problem, which still remains a combinatorially hard problem.

The MMR algorithm which we developed uses the alternating decomposition, while applying an incremental marginal return algorithm for both the weasel and strike aircraft allocation to packages. It also uses the more detailed objective function Q which arises from our ADP approaches. The algorithm is outlined below:

- Initially, allocate one strike aircraft per mission. That is, let $N_i^{Strike} = 1$ for all i.
- Set $N_i^{Weasel} = 0$ for all i. Determine weasel aircraft allocations per package as follows:
  - For each package i, compute the marginal return $MR(i)$ as follows: Let $Q_i^{+wk}\left(\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}\right)$ denote the performance achieved by adding k weasel aircraft to package i, for $k = 1, 2$. Then,
    $$MR(i) = \max_k \left\{ Q_i^{+wk}\left(\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}\right) - Q\left(\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}\right) / k \right\}$$
  - Select the package i with largest $MR(i)$, and increase $N_i^{Weasel} = N_i^{Weasel} + 1$
  - Repeat until no unallocated weasel aircraft remain.
- Set $N_i^{Strike} = 0$ for all i. Determine strike aircraft allocations per package as follows:
  - For each package i, compute the marginal return $MR(i)$ as follows: Let $Q_i^{+sk}\left(\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}\right)$ denote the performance achieved by adding k strike aircraft to package i, for $k = 1, 2$. Then,
    $$MR(i) = \max_k \left\{ Q_i^{+sk}\left(\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}\right) - Q\left(\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}\right) / k \right\}$$
  - Select the package i with largest $MR(i)$, and increase $N_i^{Strike} = N_i^{Strike} + 1$.
  - Repeat until no unallocated strike aircraft remain.
- Repeat iteration between assignment of weasel and strike aircraft until convergence or a fixed number of iterations are performed.

The above algorithm incorporates several important extensions to address the use of the objective function Q instead of the simpler model. In particular, to deal with possible regions where the function is not concave, we use two increments ($k = 1,2$) to compute the maximum marginal return. Note that the above algorithm is an approximate algorithm, as the true objective function for a multistage problem will not be separable or concave. It does provide a fast, approximate algorithm, which can be used in combination with other algorithms such as combinatorial rollout, which we consider next.

4.3.2 Combinatorial Rollout

The Combinatorial Rollout algorithm is a recent algorithm developed by Bertsekas et al [BTW97]. The basic idea of the algorithm is to improve on the performance of a baseline algorithm, which in our case is the MMR algorithm described above. These incremental improvements are related to the policy improvement step in standard policy iteration algorithms for dynamic programming.

In combinatorial rollout, we solve the optimization problem, one package at a time, as follows:

- Order the package indices $i = 1,\ldots,T$. Let the current index $i' = 1$.
- Assume that packages $\left(N_i^{Strike}, N_i^{Weasel}\right)$ are fixed for $i < i'$. Determine the package allocation to target i' as follows:
  - Enumerate all possible package allocations to target i'. For each package allocation $\left(N_i^{Strike}, N_i^{Weasel}\right)$, allocate remaining strike aircraft and weasel aircraft (not already allocated to packages $i < i'$ or allocated to i') using MMR algorithm to packages $i > i'$, and evaluate the performance of the composite assignments.
  - Select $\left(N_i^{Strike}, N_i^{Weasel}\right)$ as the allocation which gives the maximum performance in the previous substep.
- If $i' < T$, increment $i' = i' + 1$; else, the algorithm is complete.

In a setting where the performance function is evaluated exactly, combinatorial rollout is guaranteed to perform no worse than the baseline algorithm, provided the baseline algorithm satisfies a mild condition of sequential consistency [BTW97] which our MMR algorithm satisfies. However, when the performance function is evaluated only approximately, as in stochastic settings, this performance improvement is not guaranteed. In particular, the

incremental nature of the algorithm makes it difficult to distinguish among package allocations where the difference in performance is of the same order of magnitude as the evaluation error. In the next section, we describe a combinatorial algorithm, which is explicitly designed for optimization of functions with uncertainty in performance evaluation.

4.3.3 Surrogate Method

The Surrogate Method [GoCass00] is a gradient-based approach for searching the control space (i.e., sets of feasible missions). This approach constructs a continuos "surrogate" objective function that is used to generate gradient information that guides a search through the discrete space of mission allocations. It also uses a stochastic approximation technique, which allows for uncertain gradient information evaluation. The gradient information is obtained by selecting a set of neighbor points to the current allocation, and evaluating the function at these neighbor points.

The principal idea of the surrogate method is to embed the combinatorial optimization problem into a continuous optimization problem. Let $\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}$ denote the combinatorial decision variables. The surrogate method instead optimizes over variables $\left\{\left(x_i^{Strike}, x_i^{Weasel}\right)\right\}$ where x denotes a continuous allocation. Let Q denote the combinatorial performance index; the key problem is that Q is only defined on feasible nonnegative integer package assignments. The algorithm extends the function Q to a surrogate function $R\left\{\left(x_i^{Stike}, x_i^{Weasel}\right)\right\}$ defined on continuous package assignments as follows:

- Given $\left\{\left(x_i^{Strike}, x_i^{Weasel}\right)\right\}$, find the closest integer assignment $\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}$, and evaluate the performance $Q\left(\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}\right)$.
- Find $2T$ neighbors of $\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}$ by modifying the number of strike or weasel aircraft to each package one at a time by one aircraft, and evaluate the performance Q for each neighbor.
- Use the $2T+1$ values to evaluate $R\left\{\left(x_i^{Stike}, x_i^{Weasel}\right)\right\}$ as a linear interpolation of the corner values.

Note that the above approximation has $R\left\{\left(x_i^{Stike}, x_i^{Weasel}\right)\right\} = Q\left(\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}\right)$ at nonnegative integer values of the allocations $\left\{\left(x_i^{Strike}, x_i^{Weasel}\right)\right\}$. Since R is now defined over continuous variables, one can compute the gradient, which is piecewise constant over regions where the closest corner and the neighbors are constant. Note, however, that the continuous function is not differentiable at integer values, because these values are at the intersection of different piecewise linear approximations (i.e. an integer point is a corner for many regions).

The surrogate algorithm is summarized as follows:

- Initialize a feasible guess at the package allocations $\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}$ across all targets.
- Initialize the step size $a_1 = 1$, and the fractional package allocations $\{(x_i^{Strike}, x_i^{Weasel})\}$ by perturbing the integer assignments $\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}$. Initialize the iteration idex to $k = 1$.
- Perform a gradient iteration as follows:
  - Compute the $2T+1$ neighbors of the fractional package allocations $\left\{\left(x_i^{Strike}, x_i^{Weasel}\right)\right\}$, evaluate the Q function at the neighbors, and evaluate the gradient $g\left\{\left(x_i^{Strike}, x_i^{Weasel}\right)\right\}$.
  - Modify the fractional allocation as $\left\{\left(x_i^{Strike}, x_i^{Weasel}\right)\right\} = \left\{\left(x_i^{Strike}, x_i^{Weasel}\right)\right\} + a_k \cdot g\left\{\left(x_i^{Strike}, x_i^{Weasel}\right)\right\}$
  - If the new allocation $\left\{\left(x_i^{Strike}, x_i^{Weasel}\right)\right\}$ is infeasible, project it inside the feasible region by reducing each allocation by the same proportionality constant.
- Increase the iteration index $k = k+1$, reduce the step size as $a_k = 1/k$.
- Compute nearest feasible integer allocation $\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}$ and evaluate its performance.
- Repeat iterations for specified number of iterations.

Because of the piecewise linear nature of the approximation, the optimal fractional solution is at an integer value; thus, if the optimization finds the optimal allocation for the surrogate cost function $R\left\{\left(x_i^{Stike}, x_i^{Weasel}\right)\right\}$, it also finds the optimal allocation for the original objective function $Q\left(\left\{\left(N_i^{Strike}, N_i^{Weasel}\right)\right\}\right)$ [GoCass00]. Furthermore, the slowly-decreasing step size guarantees convergence to a local optimum even if the evaluations of the function Q are noisy. However, as a gradient descent algorithm, the surrogate optimization method often

converges to a local instead of a global optimum. To overcome this limitation, we implement a couple of steps: First, we initialize the algorithm with the MMR solution described previously. Second, we also perform several repetitions of the algorithm from randomly selected assignments, and select the best of the resulting allocations.

4.4 DESIGN MODEL APPROACH

In this section, we develop a design model for the system dynamics, which are describe in substantial detail in Section 3.2. Similar to the hybrid dynamical model, our design model exploits the fact that interactions between JAO objects are sparse and involve relatively few objects in each event, in order to achieve compact and efficient evaluation methods. In the following, we outline our implementation for the JAO environment.

The design model is based on a small set of dynamical models and their associated interaction dynamics. These models correspond to physical objects, such as air packages, threats, and targets. This direct mapping simplifies construction of the design model. Based on the physical objects, the design models may be distilled from the more detailed hybrid dynamical model, as follows.

Parallel composition of individual object models provides a concise representation of the JAO dynamics in the absence of interactions. When objects interact, e.g., threat and target engagements, a product composition of the associated objects concisely represents the interaction. To represent these interactions, a set of composite models representing pairs of interacting objects was developed. These models, based on individual exchanges in an underlying Markov process, capture the complex interaction dynamics, e.g., weasel suppression, target acquisition, etc. The first of these models is a transition model between an air package i and an air defense SAM j. Let π_i denote the discrete state associated with the air package, consisting of the number of aircraft of each type which remain alive in the package, and let s_j denote the state of the SAM. In our SAM models, the SAM can be in one of five states, as described in our hybrid dynamics. Given the hybrid state trajectory of the air package and the capabilities of the SAM, we compute the transition kernel $P(\pi_i(+), s_j(+) \mid \pi_i(-), s_j(-))$, which is a matrix indexed by possible package contents into and out of the engagement, and SAM status into and out of the engagement.

The second model is a transition model between an air package i attacking target k. Targets can be in one of two states, alive or dead. As before, these dynamics are lifted from the detailed hybrid model, and represented as a transition kernel between the joint states, as $P(\pi_i(+),t_k(+) \mid \pi_i(-),t_k(-))$. Other models represent independent transition dynamics in SAM states, as SAMs turn on and off, or are repaired after incurring damage. By factoring the joint probabilities at the end of each stage into products of marginal probabilities for each object, one obtains an efficient prediction of the distribution at the end of a wave of activity, which can be used to compute the performance statistics associated with any given strategy.

The above prediction approach is based on propagating through a pre-specified sequence of interaction events. An important extension that we considered in this work is closed-loop prediction, where the particular sequence of interaction events depends on the specific states that are observed as outcomes. For instance, after a specific interaction between an air package and a SAM, the air package may abort its mission if the number of surviving weasel aircraft and strike aircraft falls below required quantities. Similarly, the missions selected in the second wave of an attack depend on the relative success of the first wave missions in eliminating targets, and the number of aircraft surviving the first wave.

We focus on the problem of two-stage prediction, where information is collected at the end of a stage or wave, and the next set of missions is then adapted to the results of the first wave. Closed-loop prediction depends on the arrival of information. We assume that the state at the end of the first wave is observed, and that the strategy for the second wave is then computed. Our evaluation approach is based on computing an analytical approximation to the distribution of this state as before, sampling this distribution to generate a finite number of representative scenarios. For each sample scenario, we use a combinatorial algorithm using a single wave analytical approximation of the performance criteria to determine both the desired sequence of missions and their expected performance. The performance achieved for these samples is then averaged to obtain estimates of the two-wave performance.

Another extension that we considered was a model of partial information arrival, where ISR sensors were scheduled over the battle space. In this case, the observations are perfect, but occur over time, and are localized around the ISR sensors. The localized observations result in partial information regarding the battlespace. These observations are projected forward to the

current time using the same probabilistic transition models discussed previously. The result is a probabilistic state estimated at the decision point from which decisions must be made.

4.5 COMBINATORIAL OPTIMZATION ALGORITHMS IMPLEMENTATED

This section describes how the plant implements the different controller algorithms described in Section 4.3, including when and how they are used. Further details about each algorithm are also given. As it will be evident in the sections below, the controllers may be split up into two distinct groups depending on their function. One type is used to allocate resources (strike and weasel aircraft) at the airbase to newly constructed air package objects, which the controller outputs. In addition to configuring the air packages, the controller also sets their initial missions. The second type of controller is used to modify previously created air packages' missions while in flight. No controller has the ability to reconfigure air packages while in flight, such as moving assets between packages.

4.5.1 Retasker using Combinatorial Rollout

This controller may be called by the plant when a given TCT emerges. It has a very specific role of finding the air packages that qualify for retasking to the TCT, determining the best one (if any) to divert to it, and modifying its mission accordingly. In order for an air package to qualify for retasking, it must meet the following two criteria: 1) it must be currently flying ingress to a normal target, and 2) its ingress mission route must intersect a circle of a given radius around the TCT, as illustrated in Figure 18 below. The radius, or retask range, used in all experiments was 50 km. If an air package is tasked to an AOR, it automatically qualifies for retasking if the TCT is within the same AOR. The reason for implementing a localized retasking, via the intersection range and AOR groupings, is to avoid drastic geographic changes to an air package's mission, which would have a greater disturbance on the highly coupled missions of the other aircraft. Only allowing the packages that were already passing near the TCT to divert to it should minimize the effect on coupled activities of the previously configured packages, such as in the coordinated suppression of enemy air defense. This is very important since only one air package is diverted to the TCT, and the other packages are not retasked to account for the change in the mission queue. The Retasker was implemented in this specific, localized manner in order to provide real-time control upon TCT emergence, and also to avoid

the potential snowball effect that allowing other air packages to divert to newly unassigned targets (resulting from previous retaskings) could have.

When using the Retasker, the plant will request guidance immediately following any TCT emerge event (it is part of the event's execution logic), passing the controller the name of the TCT, in addition to the required estimated state of the world. If an air package is already assigned to the TCT, then the retask call is unnecessary and will terminate. Otherwise, the Retasker first determines which air packages are valid candidates as explained above. It then uses the combinatorial rollout algorithm discussed in Section 4.1 to find the best retask option. This is done by changing one of the valid air package's missions at a time, and evaluating the effect on the entire mission queue using the one-wave, one-stage predictor (see Section 4.6.1). The option that gives the best performance expectation is selected, and if its predicted value improves that of the original mission queue, the corresponding air package is retasked by modifying its mission and returning it to the plant as guidance.

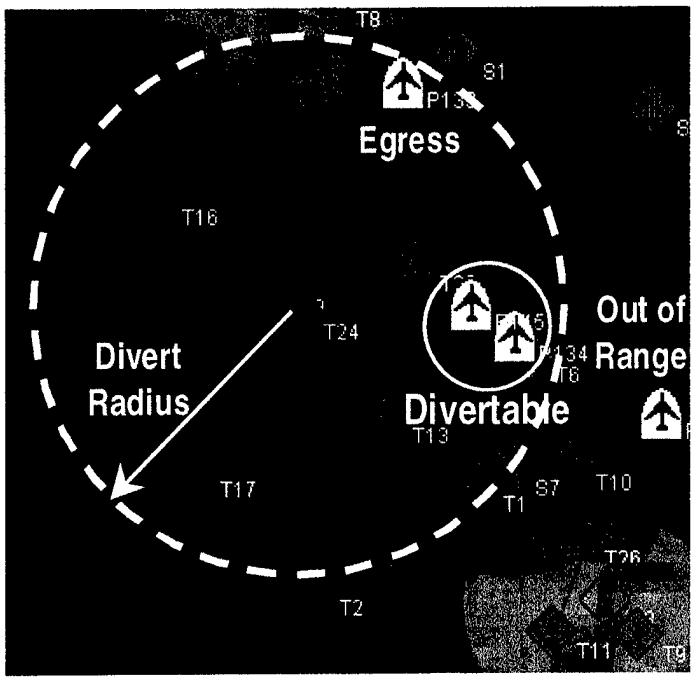


Figure 18 TCT Retasking Radius

4.5.2 Aborter using Combinatorial Rollout

This specialized controller is used in a variety of circumstances to decide whether one or more ingress air packages should abort their current missions and return to the airbase. The ability to abort missions is useful to avoid attrition to air packages, especially in an uncertain hostile environment. Depending on the current state of the world, an air package may be aborted whenever the expected gain of prosecuting its assigned target (and value of supporting other aircraft) is outweighed by the potential for further attrition. This allows the ability to save resources that might otherwise be ineffective and/or destroyed.

The Aborter functions differently depending on how the plant uses it. One way the plant may employ this controller is by calling it automatically at periodic time intervals. In this case, the Aborter performs a "full abort," giving each air package the opportunity to abort its mission, by using the combinatorial rollout algorithm discussed in Section 4.1. This works by aborting each ingress air package one at a time, evaluating the effect on the entire mission queue using the one-wave, one-stage predictor (see Section 4.6.1). The option that gives the best performance expectation is selected, and if its predicted value improves that of the current mission queue, the corresponding air package is aborted by modifying its mission to return to the airbase immediately. This process continues using the remaining air packages, evaluating each option in a mission queue that includes any previously aborted packages, until the mission queue cannot be improved by aborting any air packages. All aborted packages are then returned to the plant as guidance.

Another way the plant uses the Aborter is when a SAMSite emerges from an unknown (hiding), inactive, or repairing state. If a SAMSite activates in response to the intersection of an air package with it's range of lethality (i.e. missile range) in order to engage the package, the Aborter is called after the engagement and any possible post-engagement SAMSite transitions occur. This is useful for responding to an uncertain engagement, which may have resulted in a loss of aircraft that could compromise the success of both the attritted air package's mission and any other missions with which it is coupled. In this case, the Aborter first decides whether or not the just-engaged air package should abort its mission by comparing the respective performance expectations using the one-wave, one-stage predictor. If so, a full abort is performed (as explained above) to account for the potential effect on any other air packages. The initial fixation on the air package involved in the engagement is done to speed up the computation time, and also as an attempt to localize the abort.

If the Aborter is called in response to a SAMSite emergence that occurred stochastically, not due to a specific simulation event, the Aborter functions slightly differently. It first decides if any ingress air packages currently routed through the emerged SAMSite's range of lethality should be aborted. This is done very similarly as a full abort, except the abort candidates are limited to this subset of air packages. If any of these packages were retasked to the airbase, a full abort is then performed on the remaining air packages to account for the possible coupling across

missions. Similarly as above, the first step is performed to both minimize computation time and to localize the Aborter's effect to just those air packages involved with the specific SAMSite.

4.5.3 Target Tasking using Maximum Marginal Return

The MMR controller is used to construct and configure new air packages from resources at the airbase, and task them to targets. First, the controller must create new air packages, one for each unassigned target in the estimated world state provided by the plant. Each package is initialized with zero weasel and one strike aircraft. As detailed in Section 4.3.1, the next step is to allocate weasel aircraft to the set of packages. This is done by temporarily incrementing the number of weasels in each package one at a time, evaluating each option within the current mission queue configuration using any of the possible predictors (see Section 4.6), and permanently adding the weasel aircraft to the package with the best performance expectation. This process continues until either the airbase runs out of weasel aircraft to allocate, or adding weasels to any package does not improve the mission queue's predicted value. The algorithm may be tuned by changing the maximum number of weasel aircraft allowed in an air package, changing the increment used when allocating weasels (how many to allocate at a time), or even by stepping multiple increments ahead when searching the control space for the best predicted value. To clarify the latter option, take the example of allocating two weasels at a time. If we allow three steps into the control space, then the number of mission queues to evaluate would equal the number of air packages times three, each being incremented by two, four, or six weasel aircraft. The one option that gives the best performance expectation would have its corresponding air package's weasel count incremented by two weasels, regardless of how many were allocated when testing that option.

Next the strike aircraft are allocated using a similar process. Initially, the previous weasel assignment is untouched, but the strike aircraft in each package are cleared to zero. The strike aircraft at the base are then allocated incrementally just as the weasels were above. The only difference is in how the control space may be "stepped into." If strike aircraft were used in the previous weasel allocation example, options with four or six aircraft would only be evaluated if no options with two or four aircraft, respectively, improved the mission queue's expected performance. These tunings of the weasel and strike assignment were used to overcome specific problems where the MMR would exit prematurely before allocating many of the aircraft, depending on how the algorithm was being used. The weasel and strike allocation cycles may be

repeated as desired, although one or two iterations of the algorithm is usually sufficient. By clearing the particular type of aircraft being assigned from the current mission queue, it may be possible to improve that aircraft's allocation using the current allocation of the other types of aircraft.

Like any other controller, the plant may employ the MMR whenever it deems it useful. In the MMR's case, this would be any time air packages may be formed and tasked to targets, whether the resources are at an airbase or an AOR. Commonly, the MMR would be used at the start of the simulation and at the end of each consequent wave of attack, that is, when all air packages return to base after executing their missions. Alternately, the MMR could be called when each air package returns to base, a "gorilla" air package arrives at its AOR waypoint, or even at periodic time intervals.

4.5.4 AOR Tasking using Maximum Marginal Return

This controller is a variation of the MMR algorithm used to task air packages to targets. Its main function is still to allocate weasel and strike aircraft to air packages. But instead of tasking them to targets, their missions are to AOR waypoints, at which a target tasking controller is used. There are two ways of using this controller, depending on how many stages the user wants to model. The traditional, more accurate way of using the controller is to form a gorilla package for each AOR, and allocate resources just like in the target tasking MMR (see Section 4.5.3), but using a two-stage predictor to evaluate mission queues of gorilla air packages (see Section 4.6.5). The first stage represents the gorilla packages flying to their AOR waypoints, and the second stage is the tasking of smaller air packages from the AORs to targets and back to the airbase. The other way of using this controller is in a one-stage context. This case works just like the target tasking MMR, actually configuring air packages tasked to unassigned targets. The only difference is that they are routed through the AOR waypoint instead of directly to the target. After the aircraft allocation is complete, the aircraft in all air packages tasked to targets in each AOR are conglomerated into larger gorilla air packages tasked to the respective AORs. The only reason to use this one-stage method in lieu of using the better two-stage predictor is to save computation time. This controller is implemented by the plant at the same times as the target tasking MMR, except it can only be used to task aircraft at the airbase, not when they arrive at an AOR waypoint.

4.5.5 Target Tasking using Surrogate Method

The surrogate method is another controller used to construct and configure new air packages from resources at the airbase, and task them to targets. It is implemented by the plant in the same way as the MMR (see the end of Section 4.5.3). First, the controller must create unconfigured air packages, one for each unassigned target in the estimated world state provided by the plant. A gradient-based approach then searches the control space to allocate strike and weasel aircraft to the different packages, as detailed in Section 4.3.3. One addition to the algorithm's description above is an option that was added to speed it up. Rather than restarting the algorithm multiple times with random air package configurations (or target assignments), it was useful to seed the surrogate with the MMR's solution, and then iterate over that to try to improve upon it by moving aircraft between the packages.

4.5.6 AOR Tasking using Surrogate Method

This controller is a variation of the surrogate method used to task air packages to targets (see Section 4.3.3). Its main function is still to allocate weasel and strike aircraft to air packages. But instead of tasking them to targets, their missions are to AOR waypoints, at which a target tasking controller is used. After forming one "gorilla" package for each AOR, resources are allocated using the same gradient-based approach detailed in Section 4.3.3, but using a two-stage predictor to evaluate mission queues of gorilla air packages (see Section 4.6.5). The first stage represents the gorilla packages flying to their AOR waypoints, and the second stage is the tasking of smaller air packages from the AORs to targets and back to the airbase. This controller is implemented by the plant at the same times as the target tasking surrogate method, except it can only be used to task aircraft at the airbase, not when they arrive at an AOR waypoint.

4.5.7 Target Tasking using Combinatorial Rollout

The combinatorial rollout controller is used to construct and configure new air packages from resources at the airbase, and task them to targets, based on the algorithm in Section 4.3.2. First, the controller must create new air packages, one for each unassigned target in the estimated world state provided by the plant. The control space is then directly enumerated with every possible air package configuration to each target, taking into consideration any constraints on maximum numbers of aircraft per package and incremental assignments of aircraft (i.e. two

weasel aircraft at a time). Each option is selected and added to a mission queue to evaluate. Any resources remaining at the airbase (not in the mission queue) are used to form air packages to task to unassigned targets, using a greedy heuristic. This mission queue is then evaluated using any of the possible predictors (see Section 4.6). The air package option whose heuristically completed mission queue gives the best performance expectation is then added to the final mission queue. Any other options in the enumeration tasked to that package's target are removed from the possible candidates. In successive iterations of this option selection, the mission queue evaluated includes not only the new candidate, but also any previously selected packages, which decreases the resources available to the heuristic when filling out the rest of the mission queue. This process continues until either all resources have been exhausted, or the current mission queue's predicted value cannot be improved by adding any option to it.

This controller is implemented by the plant in the same way as the other target tasking controllers (see the end of Section 4.5.3), except it can only be used to task aircraft at the airbase, not when they arrive at an AOR waypoint.

4.6 DESIGN MODELS IMPLEMENTED

The control architecture described Section 4.2.2 executes a measured response, i.e., trading off computation and performance to specific "trigger" events. In order to achieve "closed-loop" behaviors, control decisions must explicitly account for subsequent decisions. In this section, we highlight several multi-stage controller designs, typically for the allocation of base resources, which account for subsequent decisions. In each case, what distinguishes these controllers is the associated prediction model.

4.6.1 1-Stage/1-Wave Model

The 1-Stage/1-Wave controller determines a set of missions over a single wave. Each wave begins when the air packages are launched from base and ends when they return.

This controller solves the combinatorial optimization problem discussed above (Section 4.3), and therefore any of the methods may be used. Each of these methods evaluates candidate control option using our design model (Section 4.4) over a single wave. This is depicted in Figure 19.

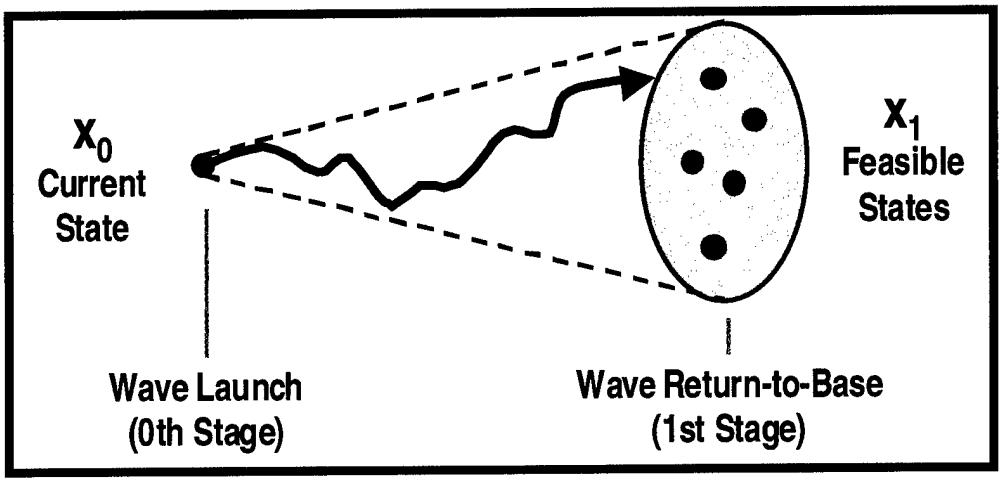


Figure 19 1-Stage Prediction

This is the baseline controller, which is also the basis for subsequent controller implementations. Due to relative performance with respect to computation, the MMR algorithm is better suited for the 1 Stage/1 Wave case.

4.6.2 2-Stage/1-Wave Model with Retasking

This controller also considers a 1-wave horizon, but accounts for potential retasking of air packages in response to emerging TCTs. TCT emergence is a random event during the execution of a wave. When a TCT emerges, divertable air packages (i.e., loaded and within range) are considered for retasking, and the best option is selected (see Section 4.5.1). Therefore, we expect the controller to assign missions from base that anticipate potential retasking. This type of proactive control significantly increases computational complexity. The difficulty is in predicting the value over 2 stages, i.e., through an intermediate decision point that corresponds to a retasking decision. We illustrate this in Figure 20. A set of missions is launched from base. When a TCT emerges, a control decision is required, which in general will map the current state x_1 at the time of the TCT emergence to an appropriate retasking control u_1. Given a specific state x_1 and control u_1, we are able to complete the 1-wave prediction. This is difficult even for a single TCT since the emergence of the TCT occurs at a random point in time and the control decision is state dependent.

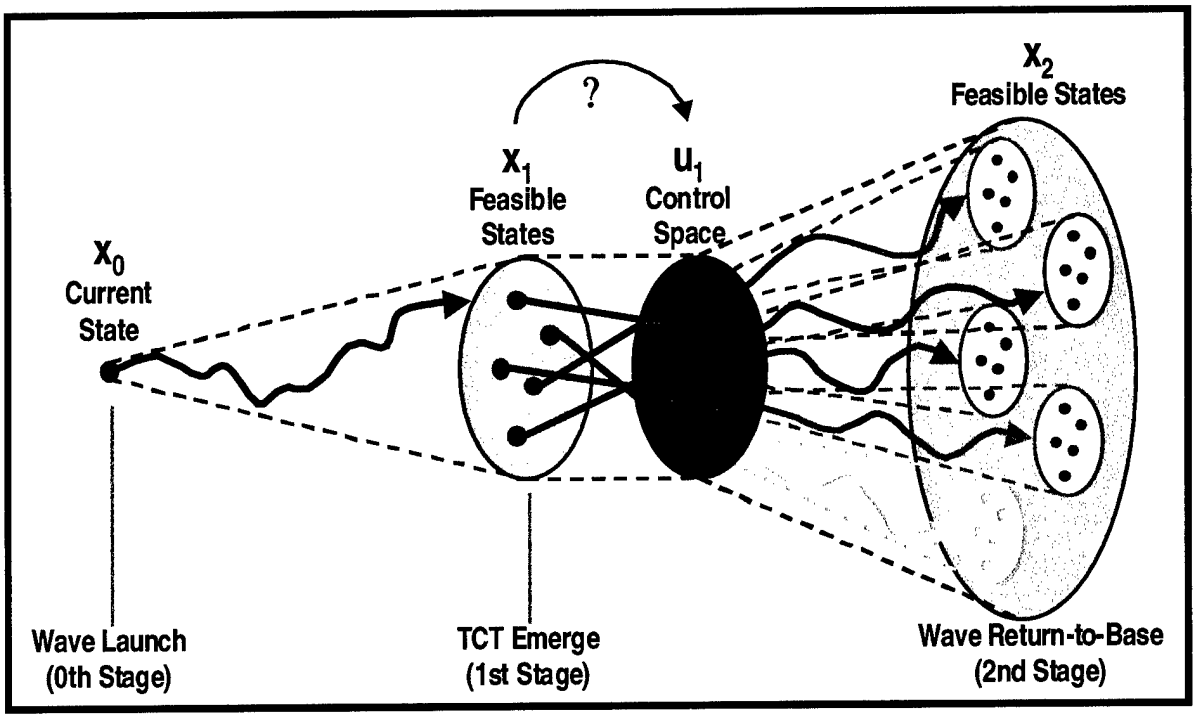


Figure 20 2-Stage Retasking Prediction

To simplify this problem, we assume that retasking depends only on the arrival probably of the TCT (rather than the precise state of all air packages, targets, and threats) and that the retask decisions are synchronized across missions. This is illustrated in Figure 21, where the first air package to enter a TCT divert range triggers a divert decision for all air packages.

With these assumptions, we are able to use the same combinatorial algorithms with a specialized predictor. Consider Figure 20, our baseline predictor is used up until the retasking point (TCT Emerge in the figure). At this point the retasking controller is used to identify a candidate retask option, which is implemented probabilistically depending on the probability that the TCT has emerged. Therefore, for a given retask option, the predicted value is given by a weighted average of either retasking or not retasking the individual air packages. Note, that the retask control algorithm needs to be solved for each evaluation of this 2-stage controller.

This controller generates an open loop control decision that hedges against the probabilistic arrival of TCTs, effectively inflating the value of missions which could be diverted while encouraging an allocation of resources appropriate for prosecution of potential TCT as well as original targets.

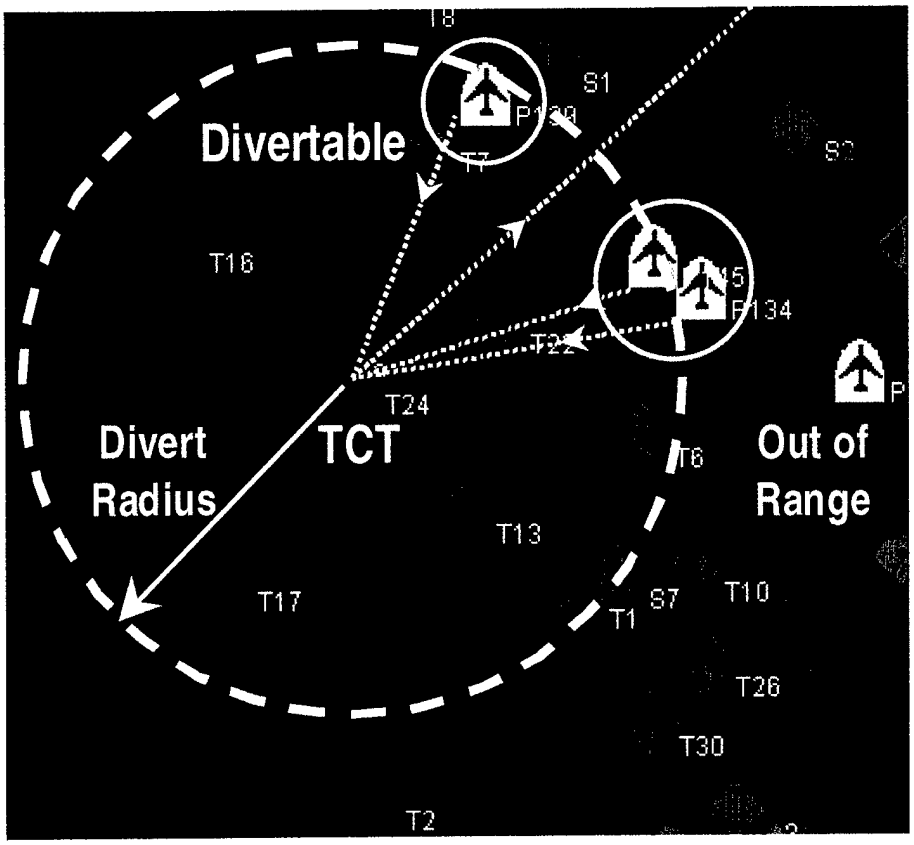


Figure 21 Retasking Approximations

4.6.3 2-Stage/2-Wave Model

In this case, we consider two waves without retasking. The controller determines a set of mission for the first wave, while explicitly accounting for second-wave control decisions. Selection of the first-wave control uses the same combinatorial algorithms as in the previous cases, however a prediction is required that considers two waves. This is depicted in Figure 22.

The set of missions is launched from base. When the air packages return to base, a control decision is required, which in general will map the current state x_1 at the beginning of the second wave to an appropriate second wave control (i.e., set of missions) u_1. Given a specific state x_1 and control u_1, we are able to compute the 2-wave prediction as a function of the x_1.

We use our baseline predictor for the first wave, which generates a probabilistic distribution $f(x_1|x_0,u_0)$ over the state x_1. The control $u_1 = \mu(x_1)$ is determined from a given state using one of the same combinatorial algorithms that were available for the first stage. To evaluate the second stage, we compute the expected value $J[f(x_2|x_0,u_0)]$ at the end of the

second stage by averaging over the state x_1 at the end of the first stage. To compute this expectation, we enumerate the feasible states x_1 at the end of the first stage, determine the second wave control $u_1 = \mu(x_1)$, and compute the value $J\left[f(x_2|x_1,u_1,x_0,u_0)\right]$ for the given state x_1. The sum of these values, weighted by the probability of the associated state x_1, provides the second stage value $J\left[f(x_2|x_0,u_0)\right]$. However, this approach is only feasible when the state space of x_1 is small. In the following, we discuss several approximations of the second stage evaluation. Specifically, we consider random sampling, certainty equivalence approximations, and an open-loop approximation.

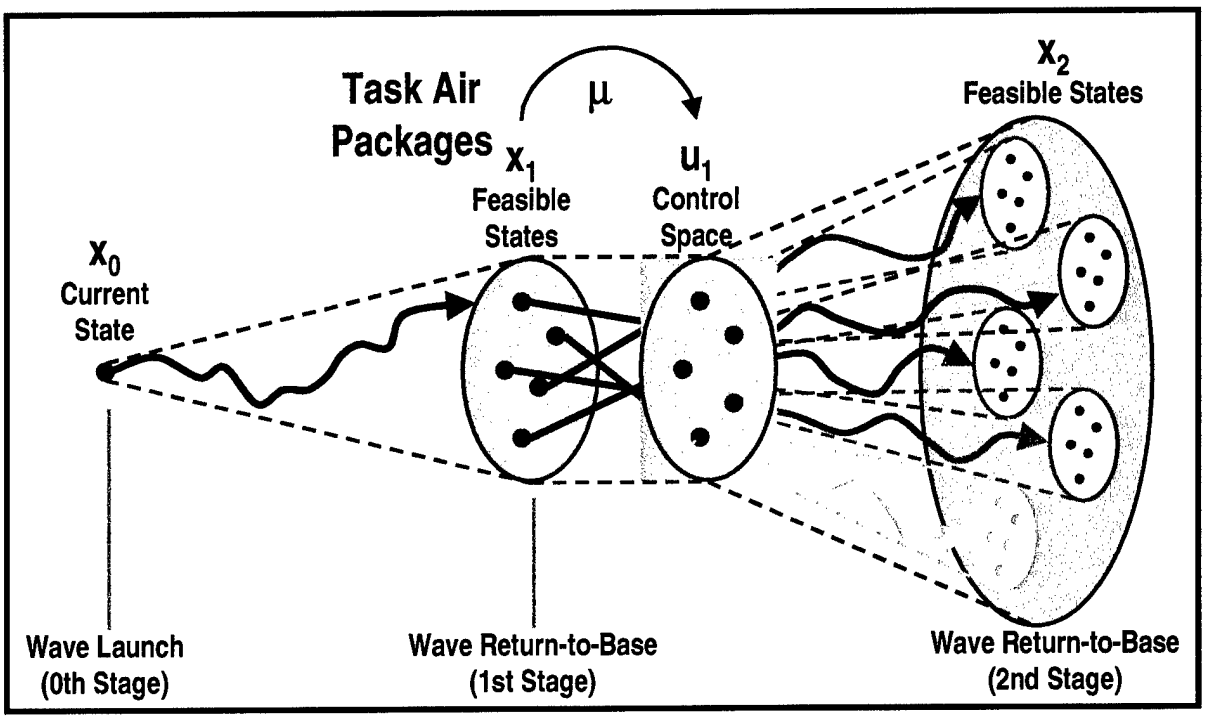


Figure 22 2-Stage/2-Wave Predictor

4.6.3.1 Random Sampling

In Figure 23, we illustrate the random sampling approach used to predict the performance of the second wave. Our design model is used to predict the value of the first wave, and it also provides the (approximate) distribution $f(x_1|x_0,u_0)$ over the state x_1 at the end of the first wave. In this approach, Monte Carlo integration is used to estimate the second stage performance $J\left[f(x_2|x_0,u_0)\right]$. A state x_1 is selected randomly according to the known distribution $f(x_1|x_0,u_0)$ at the end of the first wave. The control $u_1 = \mu_{MMR}(x_1)$ is determined based on that state x_1.

Given the state x_1 and the control u_1, our design model is used to compute the value of the second stage $J\left[f\left(x_2|x_1,u_1,x_0,u_0\right)\right]$ given the intermediate state x_1, from which we estimate the second stage performance

$$\hat{J}[f_2(x_2 \mid x_0,u_0)] = \frac{1}{N_{samples}} \sum_{i=1}^{N_{samples}} J\left[f\left(x_2|x_i,u_i,x_0,u_0\right)\right]$$

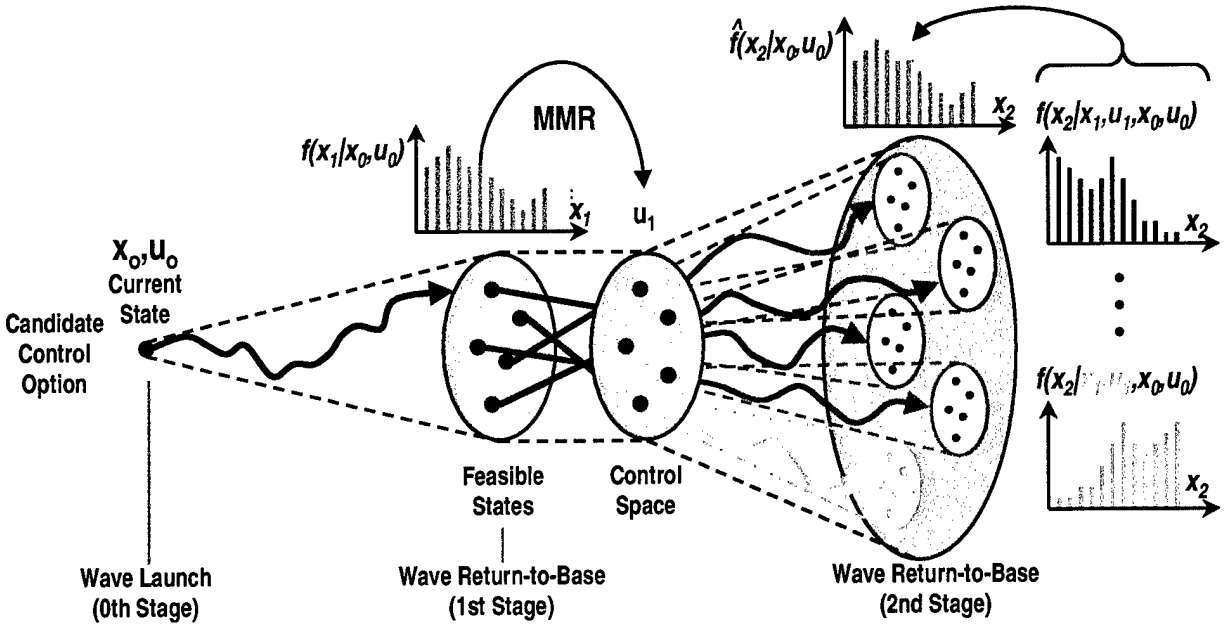


Figure 23 Random Sampling for 2-Stage/2-Wave Prediction

Using the MMR algorithm, this approach requires $O\left(N_{samples}(N_{TL}+1)^2(N_s+N_w)^2\right)$ single stage evaluations using our design model. $N_{samples}$ is the number of Monte Carlo samples. N_{TL} is the number of target locations. N_s is the total number of strike aircraft. N_w is the total number of weasel aircraft. Methods such as importance sampling could theoretically improve performance. However, preliminary experiments demonstrated only a minimal improvement in computation with comparable performance.

4.6.3.2 Certainty Equivalent Approximation

In Figure 24, we illustrate a certainty equivalent approach used to predict the performance of the second wave. Our design model is used to predict the value of the first wave and the associated (approximate) distribution $f\left(x_1|x_0,u_0\right)$ over the state x_1 at the end of the first wave. In this approach, a certainty equivalent state $\bar{x}_1$ is selected to represent the entire

distribution. In our experiments, we consider the mean and mode of the distribution. Given the certainty equivalent state $\bar{x}_1$, a control $\bar{u}_1 = \mu_{MMR}(x_1)$ is determined using the MMR algorithm. Given the state $\bar{x}_1$ and the control $\bar{u}_1$, our design model is used to compute the value of the second stage, which is our estimate of the overall second stage performance

$$\hat{J}[f(x_2 \mid x_0, u_0)] = J[f(x_2 \mid \bar{x}_1, \bar{u}_1, x_0, u_0)]$$

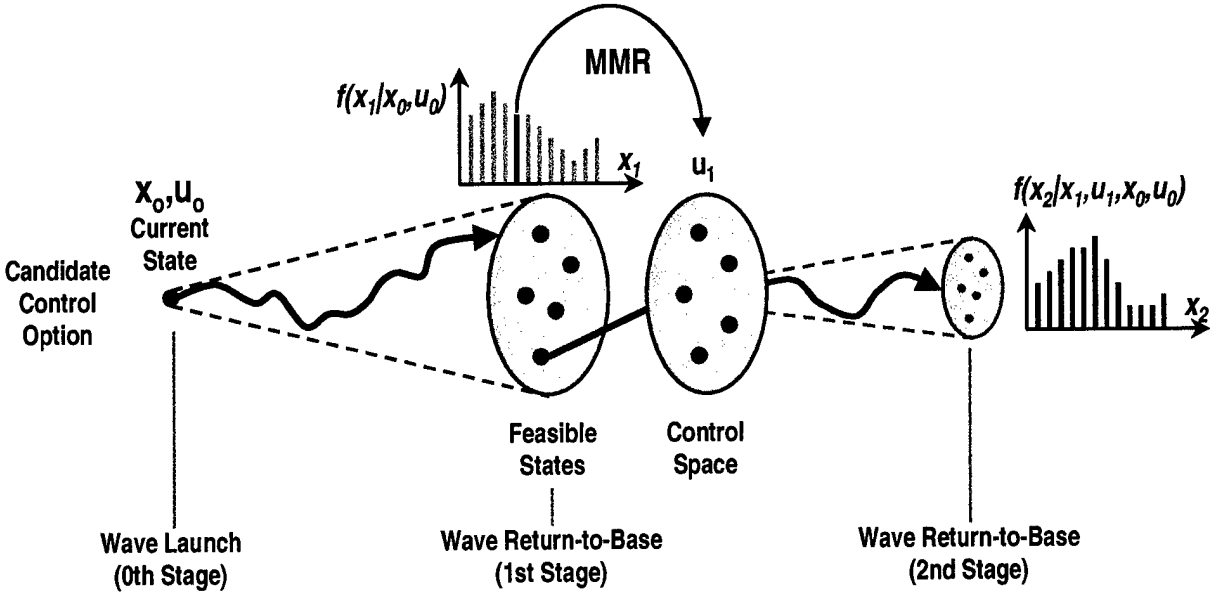


Figure 24 Certainty Equivalent Approximation for 2-Stage/2-Wave Prediction

Using the MMR algorithm, this approach requires $O\left((N_{TL}+1)^2(N_s+N_w)^2\right)$ single stage evaluations using our design model. Other certainty equivalent states could also be used in this context.

4.6.3.3 Aggregate Open-Loop Approximation

In Figure 25, we illustrate an aggregate open-loop approximation that is used to predict performance in the second wave. In this case, we augment the single stage controller by doubling the number of strike aircraft N_s that are available. The additional aircraft represent the reuse of aircraft in the second stage. Given the increased resources, we determine the initial control u_0. Due to the additional resources, this control is infeasible, i.e., there are not enough strike aircraft at base to perform all the missions. Therefore, we prune the control decision to make it feasible. Two methods are used. First, we truncate the number of missions (retaining the highest value missions) and reassign weasel aircraft. An alternate approach sends all the missions at half the

strike aircraft. Our design model is used to compute the value of the first wave $\hat{J}\left[f^{2N_s}\left(x_1^{2N_s} \mid x_0^{2N_s}, u_0^{2N_s}\right)\right]$ with the additional aircraft. The additional aircraft provide a representation of the two wave performance.

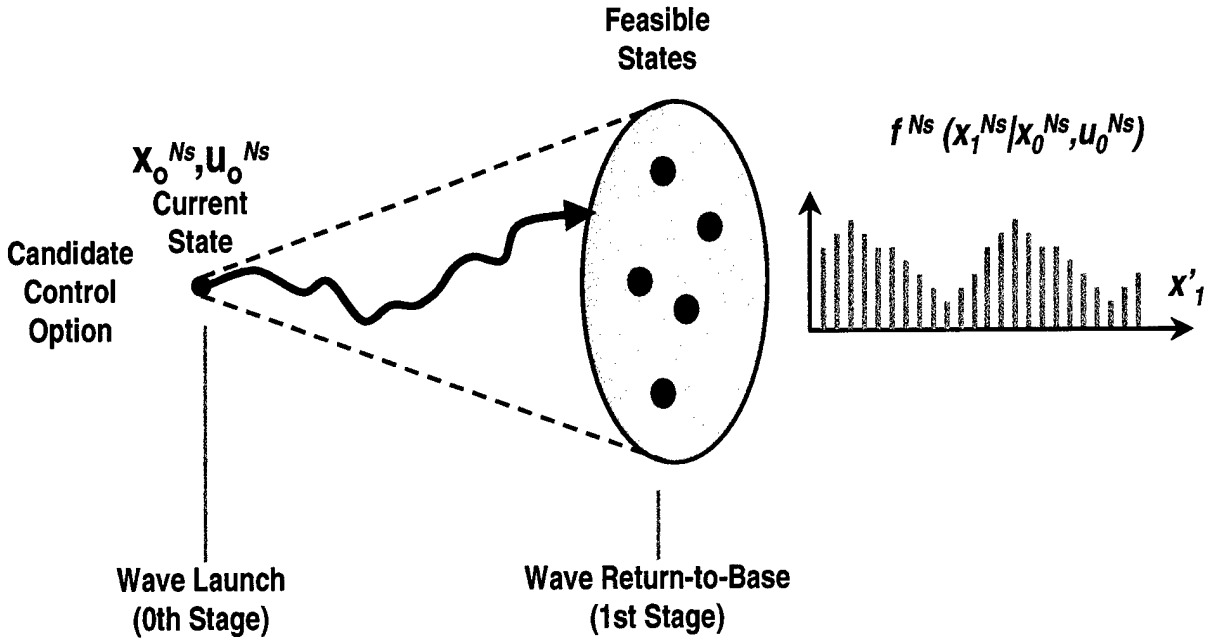


Figure 25 Aggregate Open-loop Approximation for 2-Stage/2-Wave Prediction

Using the MMR algorithm, this approach requires $O((N_{TL}+1)(2N_s+N_w))$ single stage evaluations using our design model. Similar open-loop approximation could also be used in this context.

4.6.4 4-Stage/2-Wave Model with Retasking

This controller combines the two previous controllers to further extend the control horizon. We consider two waves with retasking. The controller determines a set of missions for the first wave, while explicitly accounting for retasking in the first wave, a second-wave control decision and retasking in the second wave. Selection of the first-wave control uses the same combinatorial algorithms, however a prediction is required that considers two waves with retasking. Two of the four stages are depicted in Figure 26.

We use our retasking predictor (Figure 20) for the first wave, which results in a probabilistic distribution over the state x_2. The control $u_2 = \mu(x_2)$ is determined from a given state using the same combinatorial algorithms that were used for a single stage. To evaluate the

second stage, we average performance, again using the retasking predictor, over the state x_2. This is implemented using one of the approximation techniques discussed in the previous section, i.e., random sampling, certainty equivalence approximations, and an open-loop approximation.

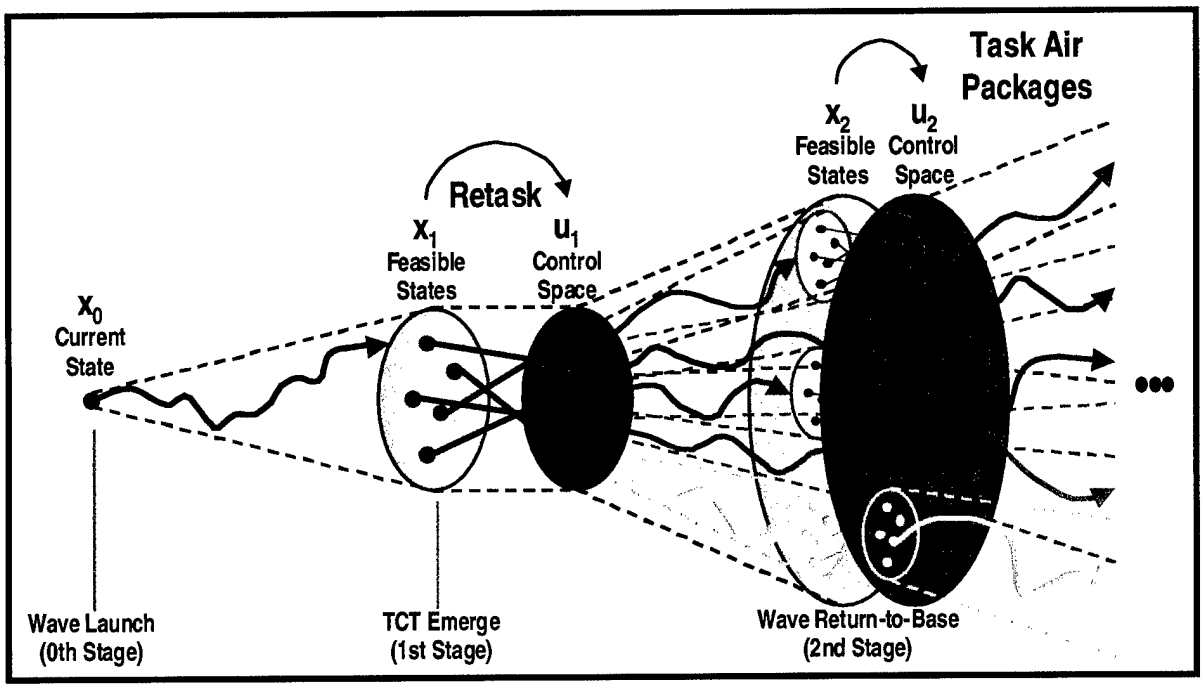


Figure 26 4-Stage/2-Wave Prediction

4.6.5 2-Stage/1-Wave Model with AOR Tasking

Given a hierarchical decomposition of the JAO environment, we formulate an alternate 2-stage problem, which first allocates ``gorrilla" air packages to specific Areas of Responsibility (AORs), and then upon arrival to the AOR, tasks regular air packages to specific targets. This effectively decomposes the 1-wave problem into smaller sub-problems associated with individual AORs. This begins to address scalability; sophisticated algorithms may be scaled to realistically sized problems or efficient algorithms may be scaled to larger problems. In this case, the base controller determines a set of missions to AOR points, while explicitly accounting for detailed tasking that occurs at the AOR points. We expect that missions from base will be constructed to provide sufficient aircraft at each AOR point to perform local tasking. This problem is complicated by the potential coupling associated with air defenses, which may occur on ingress, egress, and among AORs. We explicitly account for the ingress coupling, but to

simplify this problem we neglect egress coupling (i.e., all air packages assume full responsibility for air defenses encountered during egress) and the coupling among AORs (similarly, each AOR assumes full responsibility for air defenses in the region).

The set of missions is launched from base to specific AORs. When the air packages arrive at the AOR points, a control decision is required, which in general will map the current state x_1 to an appropriate control (i.e., set of missions within an AOR) $u_1 = \mu(x_1)$. At each AOR point, one of the combinatorial algorithms is used with a single stage predictor modified to account for fact that aircraft do not begin at base and that only a subset of targets are available. Given a specific state x_1 and control u_1, we are able to complete the 2-stage AOR prediction.

Neglecting coupling among AORs and that due to threats during egress, the base controller uses one of the combinatorial algorithms to assign aircraft to AOR point. The predictor is modified to account for the subsequent tasking of aircraft to targets when they arrive at an AOR. This is illustrated in Figure 27. Our baseline predictor is used to determine the value (lost) during ingress to the AOR points, as well as the probabilistic distribution $f(x_1|x_0,u_0)$ over the state x_1 upon arrival at the AOR points. For a given state, we are able to determine the control $u_1 = \mu_{MMR}(x_1)$, which assigns aircraft to specific targets at the AOR point. To evaluate the second stage in each AOR, we average the performance over the state x_1 similar to the 2-wave/2-stage case. The performance from each AOR is then combined to form the overall performance of the second stage. In the following, we discuss several approximations used to evaluate the second stage in each AOR. Specifically, we consider random sampling and certainty equivalence approximations as before, as well as a partial open-loop approximation.

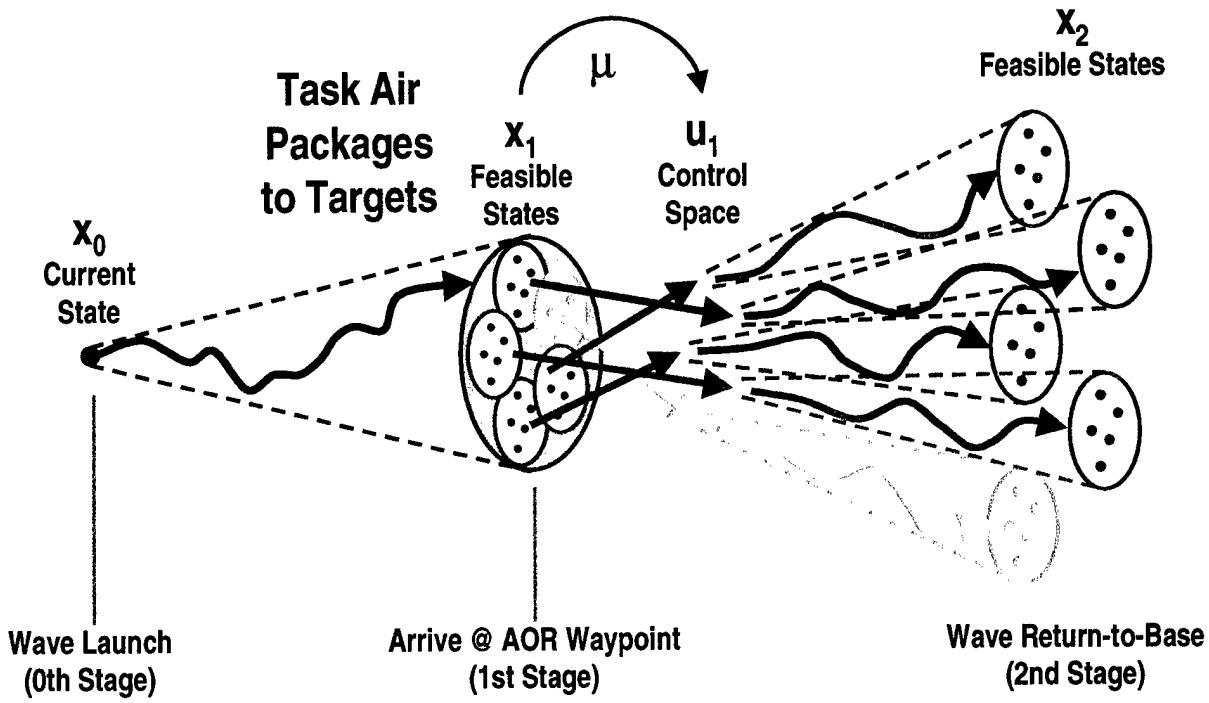


Figure 27 2-Stage AOR Predictor

4.6.5.1 Random Sampling

In Figure 28, we illustrate the random sampling approach used to predict the performance within each AOR. Our design model is used to predict the value igress to the AOR and provides the (approximate) distribution $f(x_1|x_0,u_0)$ over the state x_1. At this point, we assume that each AOR is independent. For each AOR, Monte Carlo integration is used to estimate the performance $J[f(x_2|x_0,u_0)]$. A state x_1 is selected randomly according to the known distribution $f(x_1|x_0,u_0)$ in each AOR. The control $u_1 = \mu_{MMR}(x_1)$ is determined based on that state x_1. Given the state x_1 and the control u_1, our design model is used to compute the value in each AOR, $J_i[f(x_2|x_1,u_1,x_0,u_0)]$ given the intermediate state x_1, from which we estimate the second stage performance in AOR *i*.

$$\hat{J}_i[f(x_2 \mid x_0,u_0)] = \frac{1}{N_{samples}} \sum_{i=1}^{N_{sample}} J_i[f(x_2|x_1,u_1,x_0,u_0)]$$

The overall performance of the second stage is determined by combining the value of the individual AORs.

$$\hat{J}\left[\hat{f}_2(x_2 \mid x_0, u_0)\right] = \sum_{i=1}^{N_{AORs}} \hat{J}_i\left[\hat{f}_2(x_2 \mid x_0, u_0)\right]$$

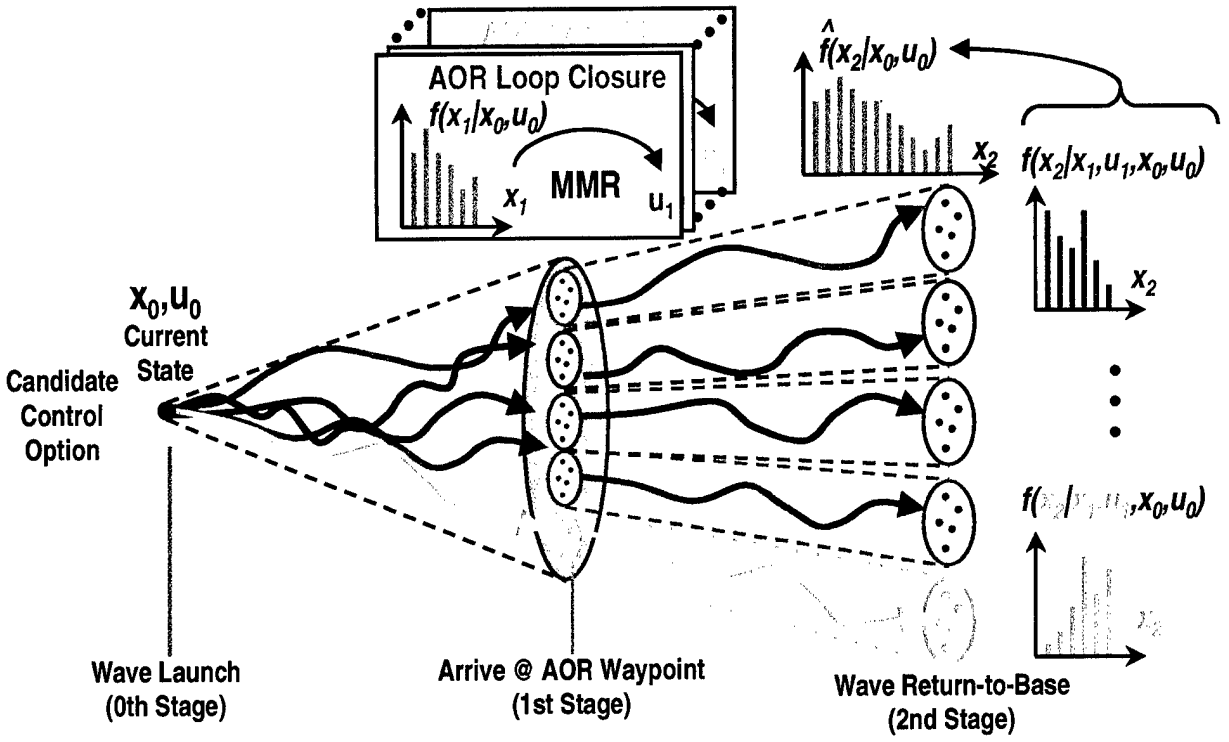


Figure 28 Random Sampling for AOR Prediction

Using the Surrogate Method as a base controller and the MMR as an AOR controller, this approach requires $O\left(N_{Asset\_Types} \cdot N_{Destinations} \cdot N_{Iterations} \cdot N_{Samples} \cdot N_{AOR} \cdot MMR_{AOR}\right)$ single stage evaluations using our design model. N_{Asset_Types} is the number of asset types, i.e., 2 in this case. $N_{Destinations}$ is the number of destinations assignable from base. $N_{Iterations}$ is the number of iterations the Surrogate Method is allowed to converge. N_{AOR} is the number of AORs. MMR_{AOR} is the computational complexity of the MMR assignment at each AOR point.

4.6.5.2 Certainty Equivalent Approximation

In Figure 29, we illustrate a certainty equivalent approach used to predict the performance within each AOR. Our design model is used to predict the value ingress to the AOR and provides the (approximate) distribution $f\left(x_1 \middle| x_0, u_0\right)$ over the state x_1. Assuming that each AOR is independent, a certainty equivalent state $\bar{x}_1$ is selected to represent the entire distribution. In our experiments, we consider the mean and mode of the distribution in each AOR. Given the

certainty equivalent state $\bar{x}_1$, a control $\bar{u}_1 = \mu_{MMR}(\bar{x}_1)$ is determined using the MMR algorithm. Given the state $\bar{x}_1$ and the control $\bar{u}_1$, our design model is used to compute the estimated value of the second stage in AOR *i*.

$$\hat{J}_i\left[f\left(x_2|x_0,u_0\right)\right] = J_i\left[f\left(x_2|x_1,u_1,x_0,u_0\right)\right]$$

The overall performance of the second stage is determined by combining the value of individual AORs.

$$\hat{J}\left[f\left(x_2|x_0,u_0\right)\right] = \sum_{i=1}^{N_{AORs}} \hat{J}_i\left[f\left(x_2|x_0,u_0\right)\right]$$

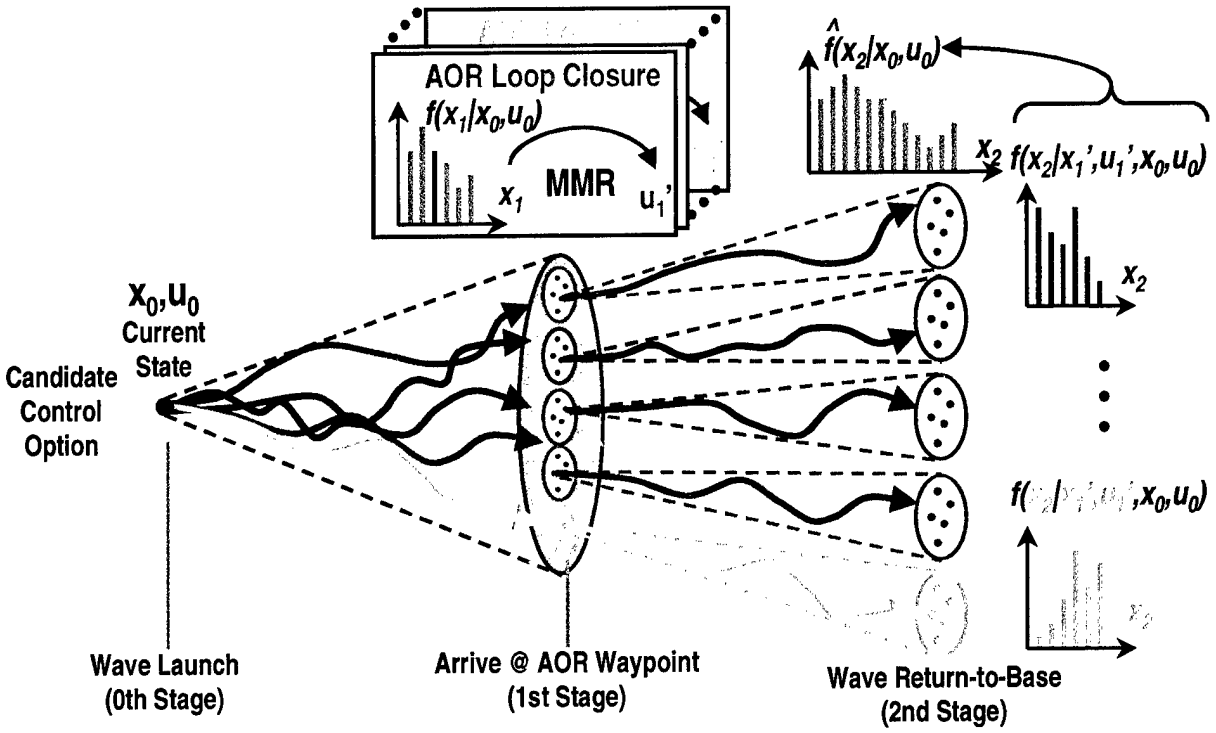


Figure 29 Certainty Equivalent Approximation for AOR Prediction

Using the Surrogate Method as a base controller and the MMR as an AOR controller, this approach requires $O\left(N_{Asset\_Types} \cdot N_{Destinations} \cdot N_{Iterations} \cdot N_{AOR} \cdot MMR_{AOR}\right)$ single stage evaluations using our design model.

4.6.5.3 Partial Open-loop Approximation

In Figure 30, we illustrate a partial open-loop approach to predict the performance within each AOR. Our design model is used to predict the value ingress to the AOR and provides the

(approximate) distribution $f(x_1|x_0,u_0)$ over the state x_1. Assuming that each AOR is independent, we consider the *k*th state of "gorilla" air package $x_1^{AP_k}$ and adopt a certainty equivalent state $\bar{x}_1^{Enemy}$ for the targets and threats.

The "gorilla" air package state $x_1^{AP_k}$ and the certainty equivalent state $\bar{x}_1^{Enemy}$ of the targets and threats are combined to establish the state x_1. A control $u_1 = \mu(x_1)$ is determined using the MMR algorithm. Our design model is used to compute the value in AOR *i*, $J_i\left[f\left(x_2|x_1^{AP_k},\bar{x}_1^{Enemy},u_1,x_0,u_0\right)\right]$ as a function of the "gorilla" air package state. The estimated value on the second stage in AOR *i* is given by

$$J_i\left[f(x_2|x_0,u_0)\right]=\sum_k J_i\left[f\left(x_2|x_1^{AP_k},\bar{x}_1^{Enemy},u_1,x_0,u_0\right)\right]\cdot P\left(x_1^{AP_k}\right)$$

The overall performance of the second stage value is determined by combine the value of individual AORs

$$\hat{J}\left[f(x_2 \mid x_0,u_0)\right]=\sum_i^N J_i\left[f(x_2 \mid x_0,u_0)\right].$$

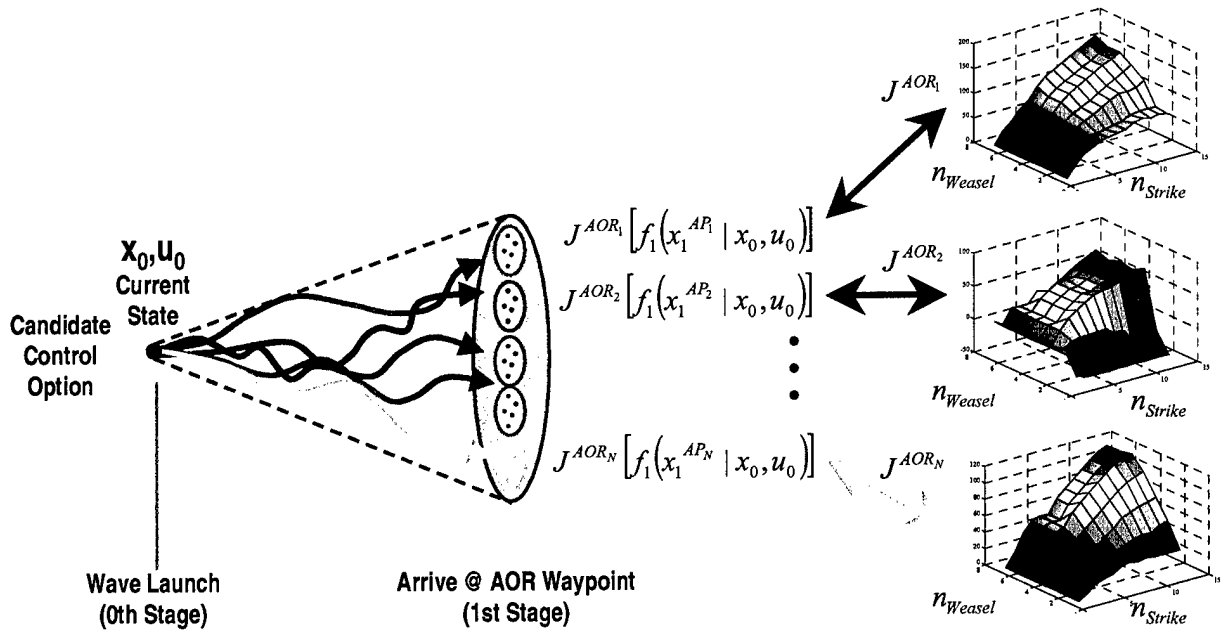


Figure 30 Partial Open-Loop Approximation for AOR Prediction

Using the Surrogate Method as a base controller and the MMR as an AOR controller, this approach requires $O(AOR_{Strike}\cdot AOR_{Weasel}\cdot N_{AOR}\cdot MMR_{AOR})$ single stage evaluations using our

design model. AOR_{Strike} is the number of strike aircraft per AOR. AOR_{Weasel} is the number of weasel aircraft per AOR.

4.6.6 2-Stage/1-Wave Model with AOR Tasking and ISR Collection

Similar to the previous AOR section, this controller allocates air packages to AORs, which are subsequently tasked to specific targets. However in this case, we consider only partial information, i.e., not all objects are observed prior to tasking within individual AORs. Similar to the previous section, this addresses scalability by decomposing the 1-wave problem into sub-problems, but also begins to address partial observations and imperfect information at the decision point.

Information dynamics describe how information evolves over time. These dynamics are captured in the probabilistic models associated with each battlespace object, as well as observation dynamics. Up to this point, all objects have been perfectly observed at each decision point. In this case, only some objects will be observed, while the probabilistic state of others evolves over time. In the absence of observations (and interactions), the probabilistic state will eventually converge to a steady state distribution which may be computed directly from the probabilistic model associated with the object. In Figure 31, we illustrate the information dynamics associated with a single object. Assuming no previous observations, the initial information state correspond to the steady state distribution $P_{Active} = 0.4$ (red), $P_{Inactive} = 0.35$ (yellow), and $P_{Unknown} = 0.25$. There is no change in the information state until an observation occurs at which point we observe the object perfectly, $P_{Active} = 1.0$. At that point (or once the object is no longer observed), information starts to degrade (i.e,. we are less certain what state the object is in). If a decision needs to be made regarding this object, we note that there is less ambiguity and thus more information during or immediately following an observation. We expect this behavior to affect the types of decisions that that the base controller makes. Namely, that aircraft should be sent to AORs that have more information (less ambiguity regarding the state of the associated objects).

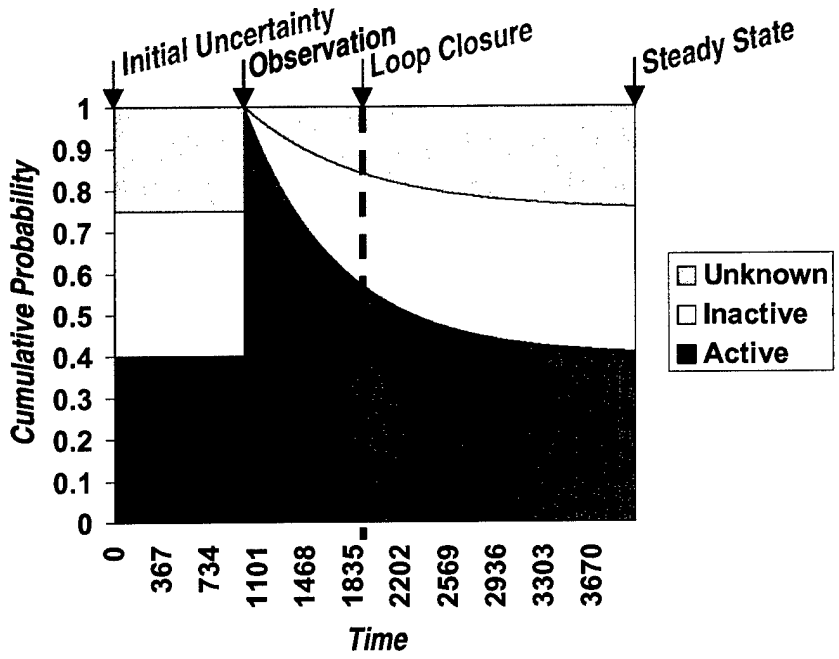


Figure 31 Information Dynamics

In this case, the AOR control problem is the same as in the previous section, except that the information available at the AOR points will be based on previous partial observations. Depending on the amount of time that has passed since the last observation, information regarding targets and air defenses will degraded. Knowing where and when observations will occur, we expect that the base controller will send aircraft to the AORs that will have better information at the associated decision point. As in the previous case, we neglect the coupling among AORs and that due to threats during egress. The base controller uses one of the combinatorial algorithms to assign aircraft to AOR points, and the predictor is modified to account for the subsequent tasking of aircraft to targets when they arrive at an AOR. This is illustrated in Figure 32. Our baseline predictor is used to determine the value during ingress to the AOR points, but also determines the distribution $f(I_1|I_0,u_0)$ from which the observations $I_1=\{z_1,z_2,\ldots\}$ are drawn. Observations that occur during ingress are projected forward (i.e., the information is degraded) to the decision point so that a current estimate $\hat{x}_1 = P(x_1|I_1)$ of the state may be used to determine the control decision $u_1 = \mu_{MMR}(\hat{x}_1)$ for the AOR. Given the control decision u_1 and the estimate of the current state $\hat{x}_1$, we evaluate the second stage in each AOR, $J_i\left[f(x_1|I_1,u_1,I_0,u_0)\right]$. We use the same methods that were used in the previous section to

estimate the second stage performance, except that the random sampling and certainty equivalent approximations are taken in terms of the observations rather than the current state. Since the partial open-loop approximation enumerates the air package state $x_1^{AP_k}$, which we assume to be perfectly observable, only certainty equivalent $\bar{x}_1^{Enemy}$ is considered in terms of the observations.

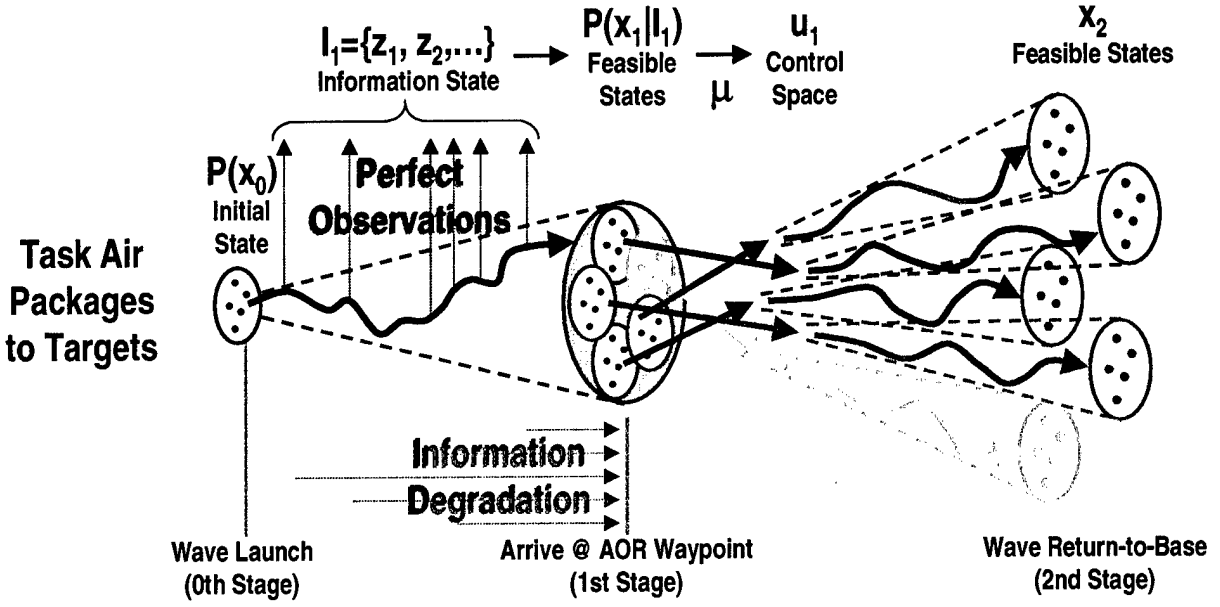


Figure 32 AOR Prediction with Partial Information

4.6.6.1 Random Sampling

In Figure 33, we illustrate the random sampling approach used to predict the performance within each AOR. Our design model is used to predict the value ingress to the AOR and provides the distribution $f(I_1|I_0,u_0)$ from which the observations $I_1=\{z_1,z_2,...\}$ are drawn. The observations are selected randomly according to the known distribution $f(I_1|I_0,u_0)$ and projected forward to the decision point, providing the current state estimate $\hat{x}_1 = P(x_1|I_1)$. Based on the state estimate $\hat{x}_1$, a control $u_1 = \mu_{MMR}(\hat{x}_1)$ is determined for each AOR. Given the state estimate $\hat{x}_1$ and the control u_1, our design model is used to compute the value in each AOR, $J_i[f(x_2|I_1,u_1,I_0,u_0)]$ given the intermediate state estimate $\hat{x}_1$, from which we estimate the second stage performance in AOR *i*.

$$\hat{J}_i[f(x_2|I_0,u_0)] = \frac{1}{N_{samples}} \sum_{i=1}^{N_{samples}} J_i[f(x_2|I_1,u_1,I_0,u_0)]$$

The overall performance of the second stage is determined by combining the value of the individual AORs.

$$\hat{J}\left[f\left(x_2|I_0,u_0\right)\right]=\sum_{i=1}^{N_{AORs}}\hat{J}_i\left[f\left(x_2|I_0,u_0\right)\right]$$

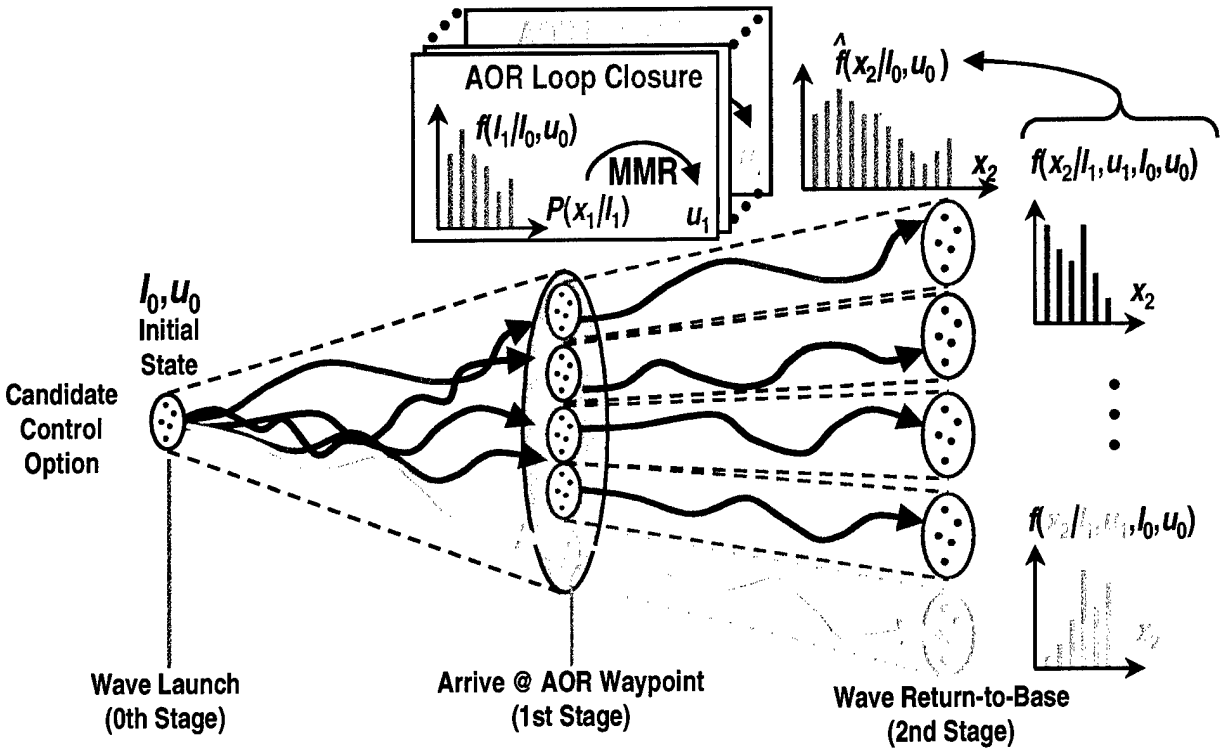


Figure 33 Random Sampling for AOR Prediction with ISR Uncertainty

Using the Surrogate Method as a base controller and the MMR as an AOR controller, this approach requires $O\left(N_{Asset\_Types}\cdot N_{Destinations}\cdot N_{Iterations}\cdot N_{Samples}\cdot N_{AOR}\cdot MMR_{AOR}\right)$ single stage evaluations using our design model.

4.6.6.2 Certainty Equivalent Approximation

In Figure 34, we illustrate the certainty equivalent approach used to predict the performance within each AOR. Our design model is used to predict the value ingress to the AOR and provides the distribution $f\left(I_1|I_0,u_0\right)$ from which the observations $I_1=\{z_1,z_2,\ldots\}$ are drawn. The certainty equivalent observation $\bar{I}_1$ is selected to represent the entire distribution and projected forward to the decision point, providing the current state estimate $\hat{x}_1=P\left(x_1|\bar{I}_1\right)$. Based

on the state estimate $\hat{x}_1$, a control $u_1 = \mu(\hat{x}_1)$ is determined for each AOR. Given the state estimate $\hat{x}_1$ and the control u_1, our design model is used to compute the value in each AOR *i*.

$$\hat{J}_i\left[f\left(x_2|I_0,u_0\right)\right] = J_i\left[f\left(x_2|\bar{I}_1,u_1,I_0,u_0\right)\right]$$

The overall performance of the second stage is determined by combining the value of the individual AORs.

$$\hat{J}\left[f\left(x_2|I_0,u_0\right)\right] = \sum_{i=1}^{N_{AORs}} \hat{J}_i\left[f\left(x_2|I_0,u_0\right)\right]$$

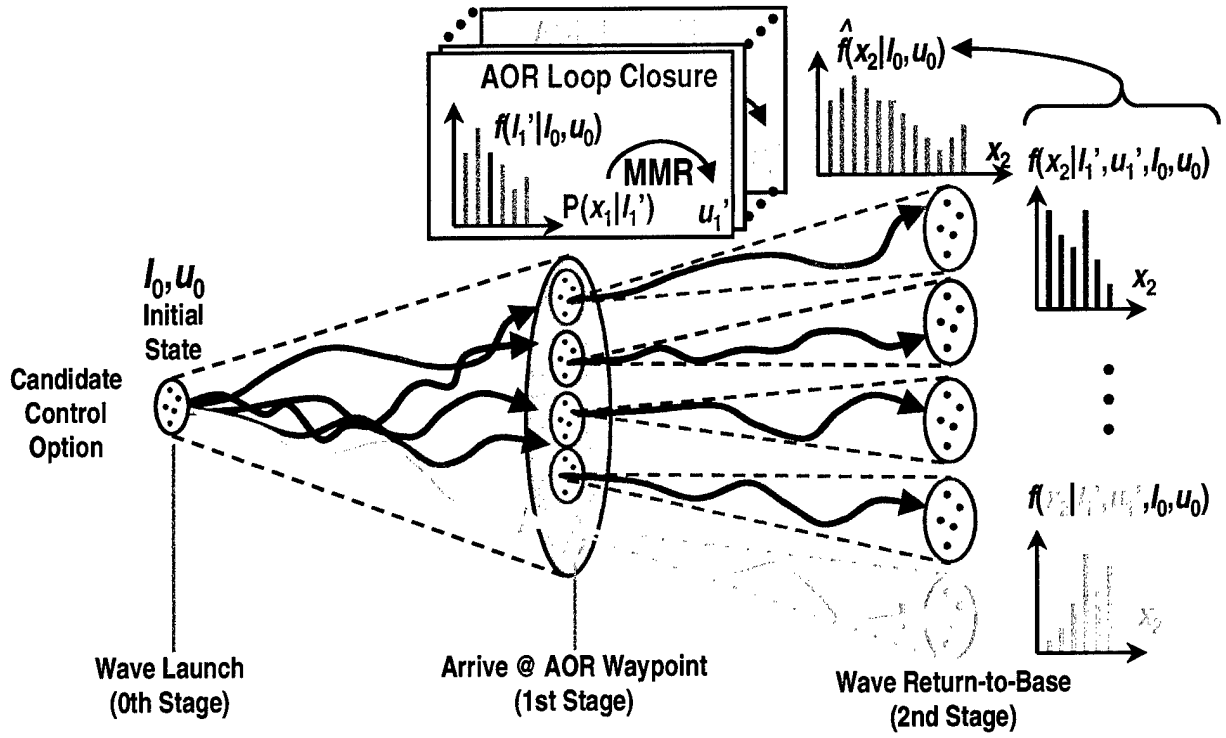


Figure 34 Certainty Equivalent Approximation for AOR Prediction with ISR Uncertainty

Using the Surrogate Method as a base controller and the MMR as an AOR controller, this approach requires $O\left(N_{Asset\_Types} \cdot N_{Destinations} \cdot N_{Iterations} \cdot N_{AOR} \cdot MMR_{AOR}\right)$ single stage evaluations using our design model.

4.6.6.3 Partial Open-loop Approximation

In Figure 35, we illustrate a partial open-loop approach to predict the performance within each AOR. Our design model is used to predict the value ingress to the AOR and provides the (approximate) distribution $f\left(I_1|I_0,u_0\right)$ from which the observations $I_1 = \{z_1, z_2, \ldots\}$ are drawn.

We consider the *k*th state of the "gorilla" air package $x_i^{AP_k}$ and adopt a certainty equivalent observation $\bar{I}_1^{Enemy}$ for the targets and threats.

The "gorilla" air package state $x_1^{AP_k}$ and the certainty equivalent observation $\bar{I}_1^{Enemy}$ of the targets and threats are combined to estimate the current state $\hat{x}_1 = P\left(x_1 \middle| x_1^{AP_k}, I_1^{Enemy}\right)$. Based on the state estimate $\hat{x}_1$, a control $u_1 = \mu(\hat{x}_1)$ is determined for each AOR. Given the state estimate $\hat{x}_1$ and the control u_1, our design model is used to compute a value in AOR *i*, $J_i\left[f\left(x_2 \middle| x_1^{AP_k}, \bar{I}_1^{Enemy}, u_1, I_0, u_0\right)\right]$ as a function of the "gorilla" air package state. The estimated value of the second stage in each AOR *i* is given by

$$\hat{J}_i\left[f\left(x_2 \middle| I_0, u_0\right)\right] = \sum_k J_i\left[f\left(x_2 \middle| x_1^{AP_k}, \bar{I}_1^{Enemy}, u_1, I_0, u_0\right)\right] \cdot P\left(x_1^{AP_k}\right)$$

The overall performance of the second stage value is determined by combining the value of individual AORs

$$\hat{J}\left[f\left(x_2 \mid I_0, u_0\right)\right] = \sum_{i=1}^{N_{AORs}} \hat{J}_i\left[f\left(x_2 \mid I_0, u_0\right)\right].$$

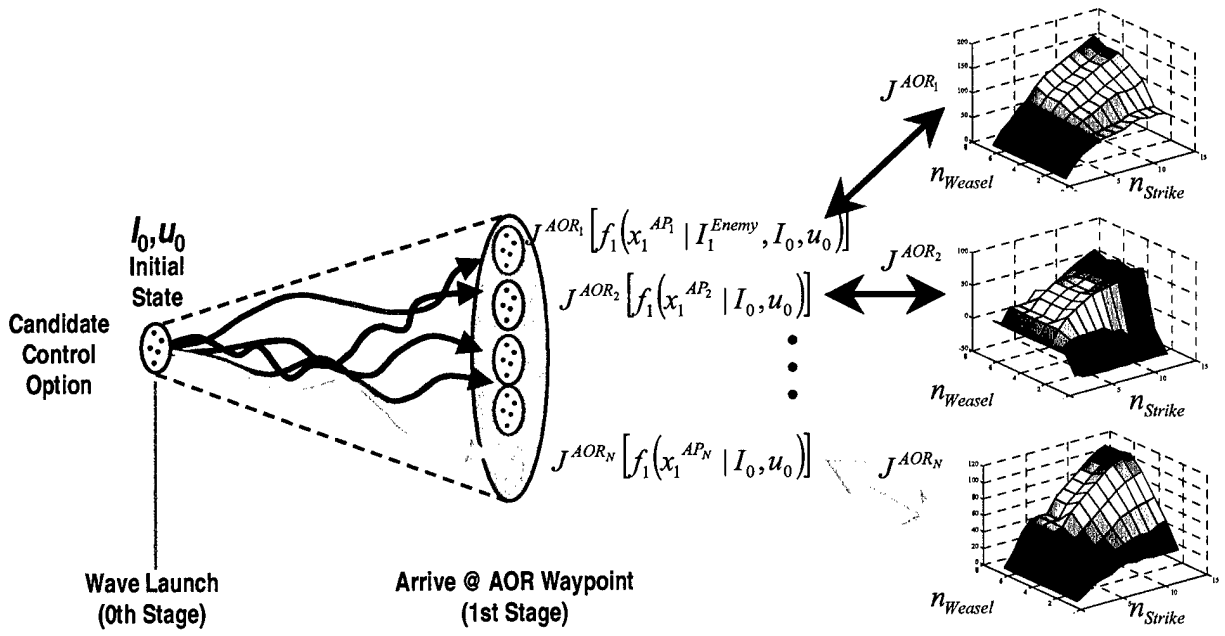


Figure 35 Partial Open-loop Approximation for AOR Prediction with ISR Uncertainty

Using the Surrogate Method as a base controller and MMR as an AOR controller, this approach requires $O\left(AOR_{Strike} \cdot AOR_{Weasel} \cdot N_{AOR} \cdot MMR_{AOR}\right)$ single stage evaluations using our design model.

4.7 JAO SCALABILITY ASSESSMENT

As presented in Section 4.2, the goal of this research was to develop ADP algorithms that produce operationally consistent behaviors for realistic sized JAO scenarios. One immediate complexity reduction was achieved by adopting a hybrid, multi-rate control architecture that tailors the application of control, i.e. reactive or proactive, for the battlespace situation at hand; however, additional complexity reduction was required. Thus, the research was focused on developing fast and efficient combinatorial assignment and prediction models that form the foundation of the ADP algorithm. In this end, it was the goal of this research to develop a spectrum of ADP control strategies that can be mixed and matched to provided a broad range of performance and computational complexity. This was achieved. In the previous four sections, the details of the efficient combinatorial assignment algorithms along with the fast analytic prediction models were presented. Accordingly, Figure 36 summaries the ADP algorithms that were developed and implemented as part of this research. Again, depending on the battlespace situation, different assignment algorithms and prediction models can be combined to produce a tailored application of control.

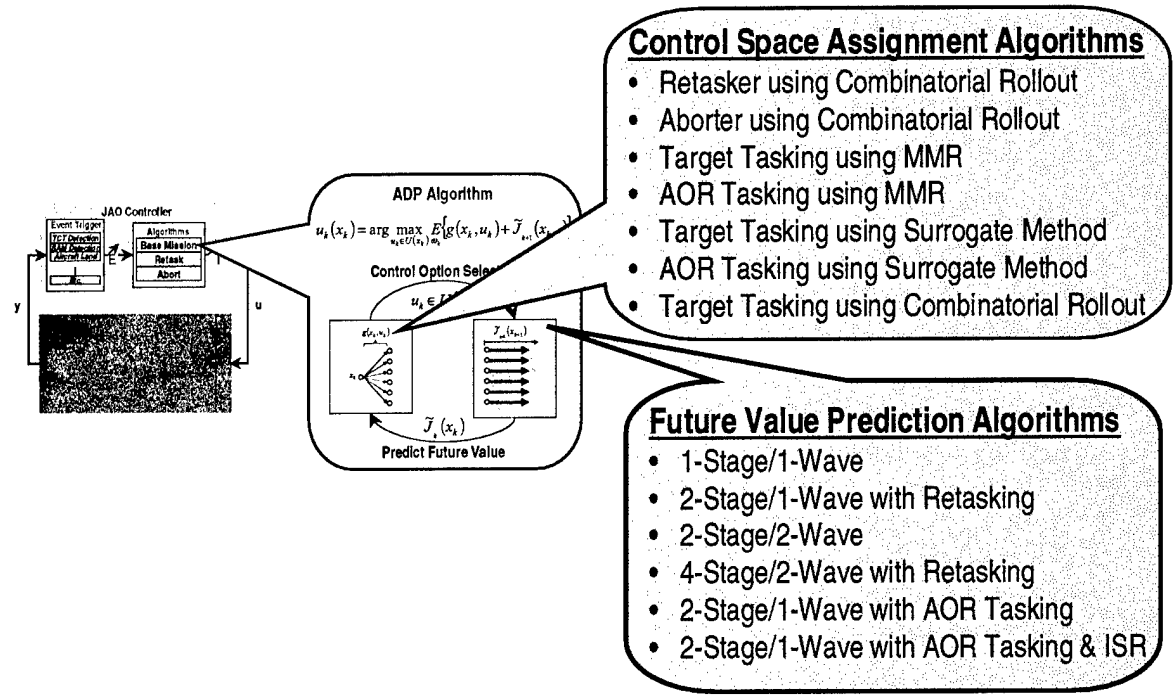


Figure 36 ADP Algorithms Developed and Implemented in Hybrid, Multi-Rate Control Architecture

Having developed and implemented these algorithms, the question remains. Do these ADP algorithms produce proactive, operationally consistent behaviors in real-time or near real-time for realistic sized JAO scenarios? The behavioral part of this question is the topic of the next chapter. Here we present the scalability assessment for the different ADP combinations implemented for generating a base mission queue without AOR tasking. Figure 37 presents the computation complexity of the base mission controllers which represent an upper bound on the computation complexity of the ADP implemented in ALPHATECH's BMC3 Development

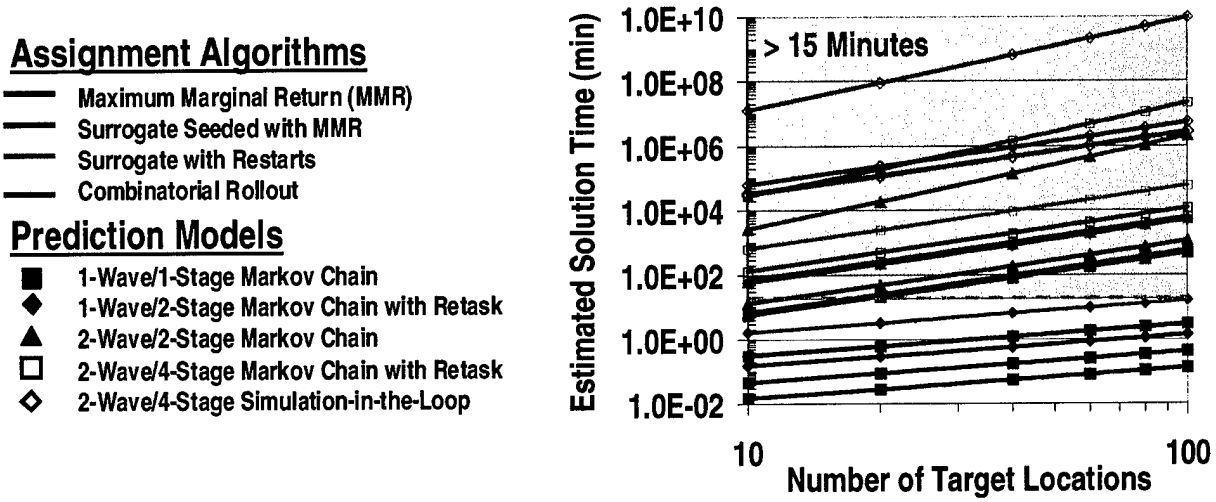


Figure 37 Scalability Assessment of ADP Algorithms Developed and Implemented for Hybrid, Multi-Rate Control Architecture (Single CPU)

Environment. Note, the shaded area represents control solutions that require more than 15 minutes to compute. Furthermore, a 100-target scenario corresponds to approximately 30 target locations.

The assumptions for this assessment are as follows:

- Base Mission Controller Generating Mission Queue
- Base Resources: 24 Strike and 12 Weasel Aircraft
- Maximum Package (6 Strike, 2 Weasel)
- 3.57 DMPIs per Target Location
- MMR for Future Base Loop Closures
- CE Analytical Predictors
- Performance on 850 MHz CPU
- Distributed on 125 CPU Array

It is seen from this figure that there is true a spectrum of solution approaches in terms of computation complexity. Figure 38 illustrates the computation complexity if is assumed that the AOC has distributed computational capability.

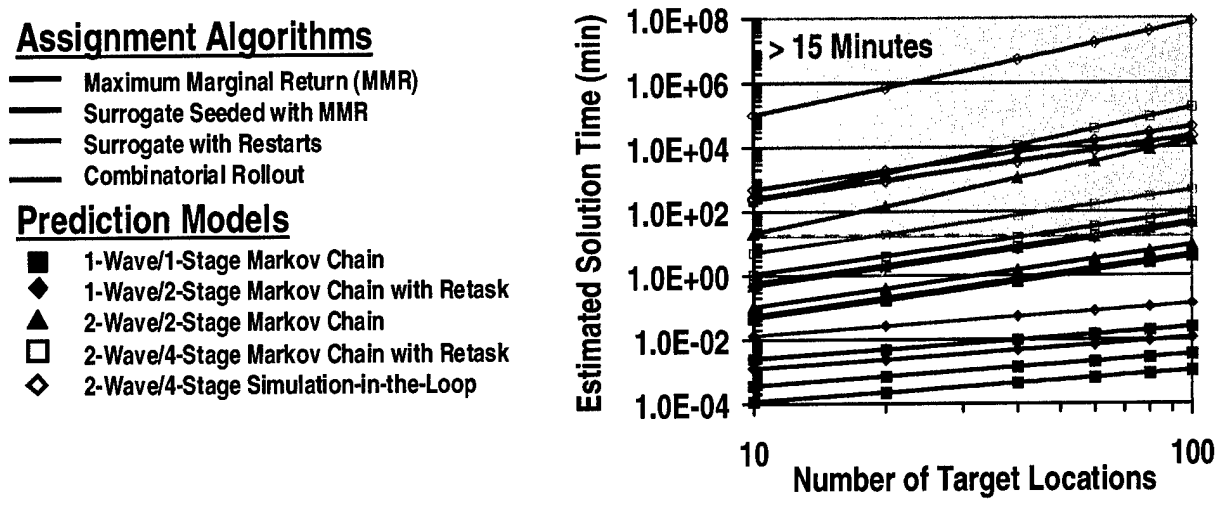


Figure 38 Scalability Assessment of ADP Algorithms Developed and Implemented for Hybrid, Multi-Rate Control Architecture (125 CPUs)

Thus, it is seen from the two figures above that there is a variety of base mission controller that can generate mission queues in near real-time for realistic sized JAO scenarios. In particular, the distributed MMR with analytical 4-stage/2-wave predictors provides near real-time computation performance, for scenarios with approximately 60 target locations. Likewise, the distributed Surrogate Method with analytical 2-stage/2-wave predictors provides near real-time computation performance, for scenarios with approximately 60 target locations.

In summary, this scalability assessment illustrated that real time or near real-time computational performance is achievable for most of the ADP algorithms developed and implemented as part of this research. Furthermore, this scalability assessment indicated that many of the ADP controllers could provide near real-time performance for scenarios with 250 targets if some modest parallel computation capability was available in an AOC.

5 EXPERIMENTATION RESULTS

In this section, experimental results that illustrate proactive, operationally consistent control strategies for the ADP algorithms developed in the previous section will be presented. The primary emphasis of this experimentation is to demonstrate the benefits of proactive versus reactive control strategies for relevant JAO problems. In this end, control behaviors and empirical simulation results will be presented to highlight the differences between the two control paradigms. All experimental results were generated using ALPHATECH's BMC^3 Development Environment.

The presentation of these experimental results is organized such that we begin with simple JAO problems and then build upon them in both terms of scenario complexity and controller complexity.

5.1 DEMONSTRATION SCENARIO

The scenario used for this assessment is based on the Cyberland Scenario provided by the DARPA/JFACC Program Office. For this scenario, it has been assumed that a higher level of decomposition, both spatial and temporal, of the JFACC objectives has been performed. The enclosed scenario represents a 24-hour segment of the air campaign and the Area of Responsibility (AOR) is the Northern Air Defense (AD) District in West Cyberland. Key features of this scenario include approximately 100 targets and threats and 36 air vehicle assets of which there are 24 generic strike aircraft and 12 generic weasel aircraft. As noted in Section 3, targets may be known, unknown (hidden), time critical, or destroyed. Threats may be active, inactive, unknown, repairing, or destroyed. There is also uncertainty in the interaction between enemy and friendly assets that depends on the characteristics of the target or threat and the composition of the air package involved, i.e. the number of strike and weasel aircraft, and also on the geometry of the interaction.

Figure 39 illustrates the target/threat laydown used for this experimental evaluation. It is seen from this figure that there are 24 normal target locations, which are represented by ◆. A normal target is a generic target that is known for all time and has 4 Designated Mean Points of Impact (DMPIs). The scenario also contains 4 TCT regions, each containing 2 emerge locations. In this context, TCTs are static, and are characterized by a single DMPI per emerge location. The TCTs emerge and hide based on stochastic processes; through the combination of temporal

and event-based stochastic transitions, the TCTs are on average only vulnerable for approximately 15 minutes in this scenario. When the TCT is hiding, it is represented by ◆, and when vulnerable, it is represented by ◆. It is also seen from this figure that the scenario contains 13 SAM sites of varying size. The position of the SAM site in this scenario is represented by ◆, and the ring around the SAM represents its lethal range. The color of this ring represents the status of the SAM sites; the color scheme is as follows: red represents radar on, yellow represents radar off, green represents under repair, and black represents unknown.

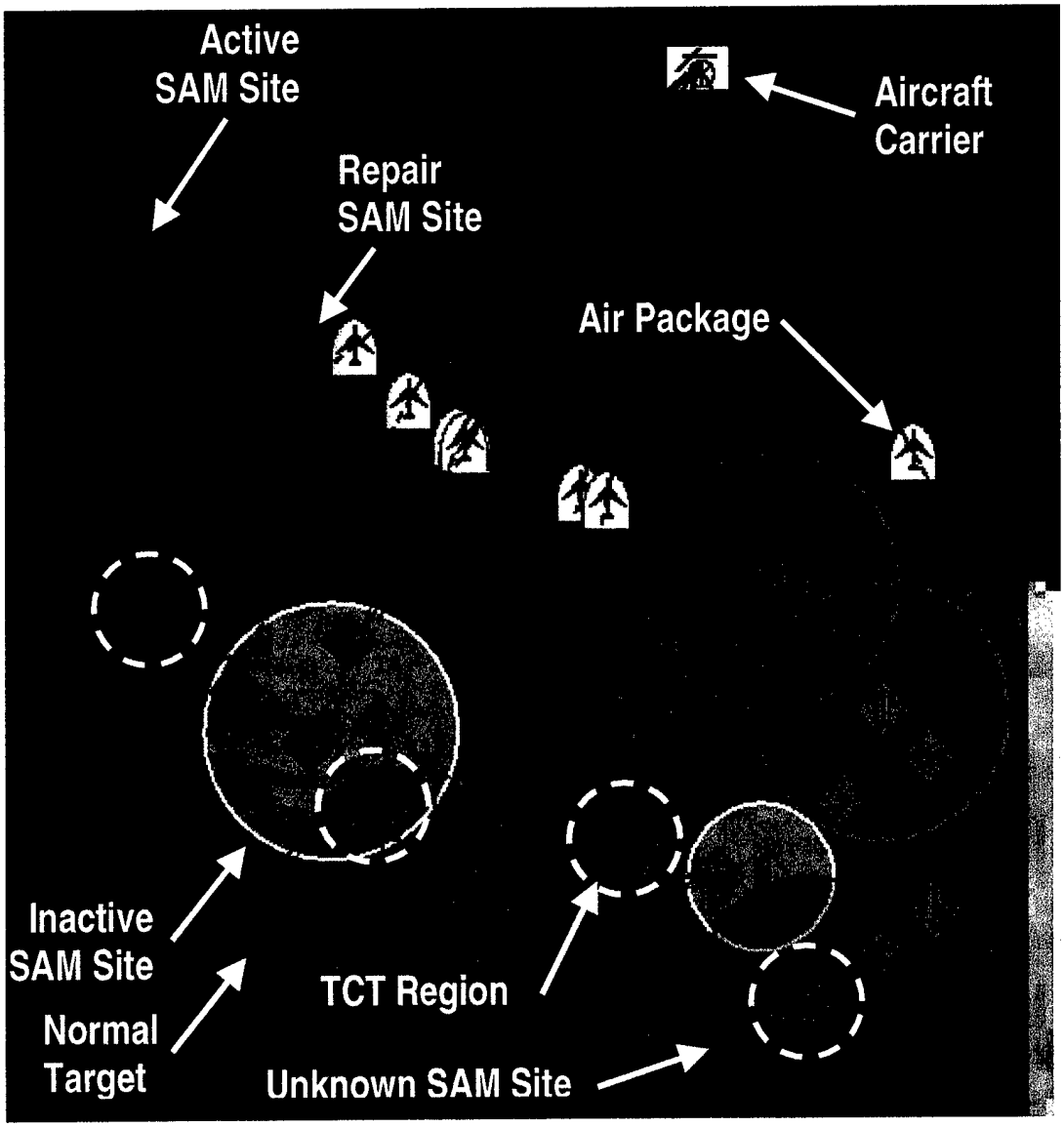


Figure 39 Demonstration Scenario Used for Experimental Evaluation

In terms of the friendly assets, there is an aircraft carrier positioned off the northern coast, which contains half of a squadron of generic strike and weasel aircraft available for this JFACC objective. There is a total of 24 strike aircraft and 8 weasel aircraft. As mentioned in the previous section, the controller composes and tasks air packages to target locations. In this scenario, an air package is represented by  , and its mission is denoted by ▬. It is assumed in this scenario that the maximum air package size is 6 strike and 2 weasels. It is also assumed that aircraft are assigned to air packages in increments of 2. Given the air package composition and tasking, the controller has the capacity to define a risk avoidance route. For some experiments, this route is determined *a priori*, and in others, the route selection is part of the control space. Finally, it is assumed in this scenario that air vehicles do not have adequate range to circumvent this defense posture.

Finally, as noted in Section 4, a relative valuation scheme is required to distinguish control options. For this scenario, all normal targets are valued at 40 points and TCTs are valued at 400 points. In terms of the airborne assets, both strike and weasel aircraft are valued at 40 points each. Note, SAM sites have no explicit value, however, they clearly have implicit value given that they affect the attrition of aircraft. Given this valuation scheme, the performance metric used for the control optimization is the sum of the target value destroyed minus the aircraft lost.

In summary, Figure 39 illustrates the baseline scenario that was used for a series of experiments that illustrate proactive, near real-time control performance of the controllers presented in Section 4. Again, the presentation of these experimental results is organized such that we begin with simple JAO problems and then build upon them in both terms of scenario complexity controller complexity. As the complexity increases, minor changes to the baseline scenario were required and will be called out in the appropriate sections.

5.2 1-STAGE/1-WAVE PROBLEM

The first set of experiments were performed to assess the accuracy of the analytic predictors over a set of control decisions. The details of this analytic prediction model are contained in Section 4.6.1. To perform this assessment, a one-wave problem was set up where the initial mission queue was defined *a priori* and no loop closures were permitted during the execution of the wave, i.e. no retasks or aborts. In this situation, the wave begins when all air

packages are launched at time $t=0$, and the wave ends when all air packages return to base. Results will be presented for the cases when the initial status of the enemy assets is either known or is specified by a distribution. Finally, straight line routing was used for all air packages since this condition results in higher attrition, and thus will highlight the prediction accuracy of the baseline analytic prediction model presented in Section 4.6.1.

The first experiment that was run to make this assessment was for the case when the initial status of the enemy assets was know *a priori.* This situation is illustrated in Figure 40 where the initial state is known at t=0, and the battlespace state is propagated to the wave return-to-base. Using the prediction model presented in Section 4.6.1, the estimated performance of the air package assignment was obtained and compared to simulation runs. With 10,000 Monte Carlo performance runs, the performance prediction using the 1-stage/1-wave prediction model was statistically equivalent to the simulated performance.

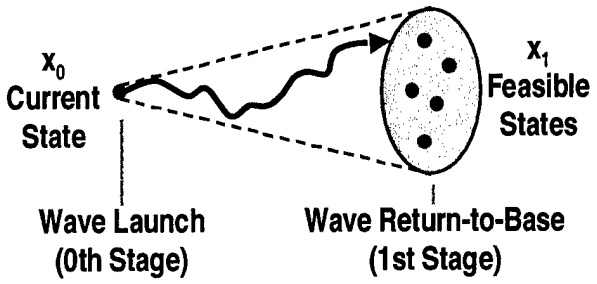


Figure 40 Battlespace State Propagation Diagram for Known Initial State

Given the experimentally demonstrated accuracy of the 1-stage prediction model given a known initial state, the next step was to assess the accuracy of the prediction model for an uncertain initial state of the enemy. As discussed in Section 4.6, many of the multiple state prediction models require the propagation of the battlespace state given uncertain enemy status. As in the previous evaluation, the air package composition and tasking is specified as part of the initial state x_o. Figure 41 illustrates the 1-stage prediction problem for the case where the initial state is defined by a distribution. Given the air packages and enemy status distributions at t=0, the battlespace state was propagated to the return-to-base without any retasking or aborts. Based on the return-to-base distribution obtained from the prediction model, the expected performance was computed. For the empirical evaluation, 2,000,000 Monte Carlo evaluations were obtained by first generating 200 realizations of enemy status at t=0 and then for each sample, obtaining 10,000 samples of the

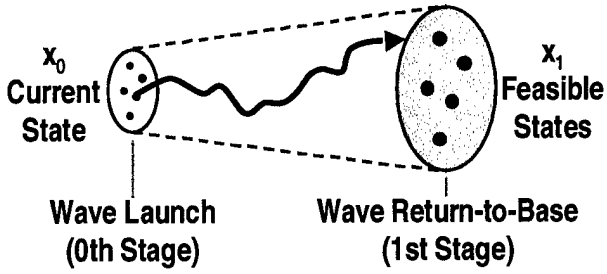


Figure 41 Battlespace State Propagation Diagram for Unknown Initial State

expected performance. Based on these evaluations, it was determined that the 1-stage prediction model was statistically optimistic by 6%.

5.3 2-STAGE/1-WAVE WITH RETASKING PROBLEM

The next set of experiments were performed to assess the performance of the 2-Stage/1-Wave Retasking model in both terms of prediction quality and control solution quality. The details of this analytic prediction model are contained in Section 4.6.2. For this 1-wave problem, all air packages launch at $t=0$, and when a TCT emerges, the retasker controller outlined in Section 4.5.1 is called to divert ingress air packages to the TCT. As before, the wave ends when all air packages return to base. As noted in Section 4.6.2, the difficulties of modeling the future retasking is the loop closure is dependent on the TCT emerge event which is stochastic based and multiple TCT emerge events can occur during a single wave. The results of this assessment will be presented below.

To perform the prediction accuracy assessment, the one-wave problem was set up where the initial mission queue was defined *a priori* and this mission queue was executed. Figure 48 illustrates the 2-Stage/1-Wave retasking prediction problem where the initial state includes the initial mission queue and the enemy status at $t=0$. Results will be presented for the cases when the initial status of the enemy assets is either known or is specified by a distribution. To perform this assessment, the analytic model with retasking was compared to experimental evaluation and to the prediction model that does not model the retask loop closure, i.e. reactive control strategy.

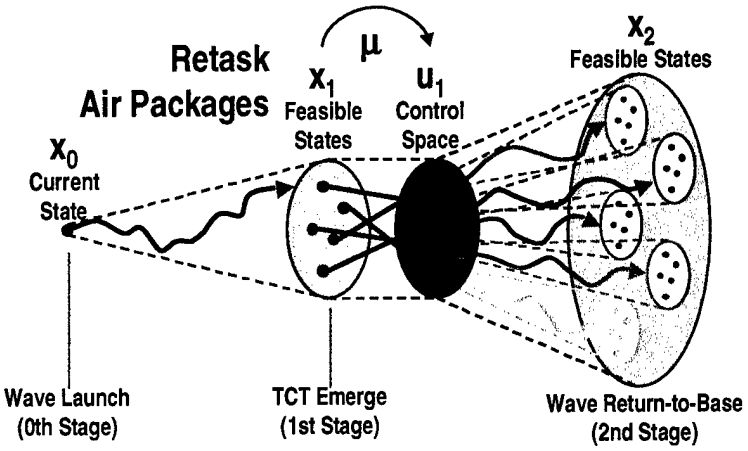


Figure 42 Battlespace State Propagation for 2-Stage/1-Wave Retasking Problem

Figure 43 illustrates the prediction accuracy of the two analytic prediction models compared to empirical evaluation. For the empirical evaluation, 10,000 Monte Carlo evaluations were performed to generate the empirical performance prediction. It is seen from the figure that the analytic model that accounts for the retasking loop closure is within 11% of the true expected performance. Furthermore, the analytic model that does not account for the retasking loop closure is only within 50% of the true expected performance.

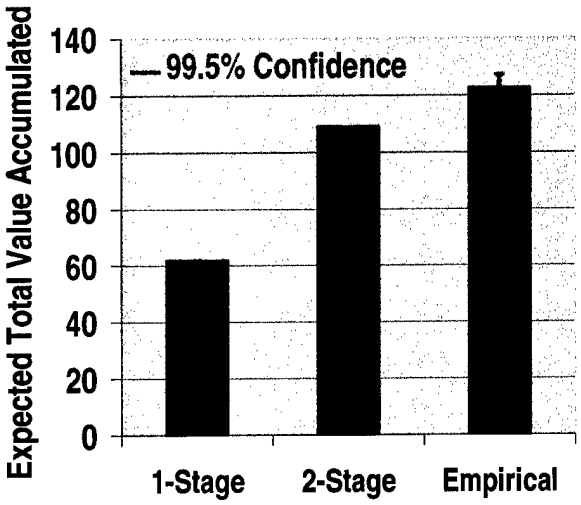


Figure 43 Prediction Accuracy of Different Design Models for the 2-Stage/1-Wave Retasking Problem with Known Initial State x_0

Next, the prediction accuracy assessment was performed for the situation where the initial status of the enemy is uncertain and is specified by distributions. Given the air packages and enemy status distributions at t=0, the battlespace state was propagated to the return-to-base with retask loop closures for TCT emerge events. As for the known enemy status case, results were obtained from the 2-Stage/1-Wave Retasking Predictor, 1-Stage/1-Wave Predictor, and experimental evaluation. For the empirical evaluation, 33,500 Monte Carlo evaluations were obtained by first generating 335 realizations of enemy status at t=0 and then for each sample, obtaining 1,000 samples of the expected performance. Figure 44 illustrates the prediction accuracy of the two analytic prediction models compared to the empirical results. It is seen from

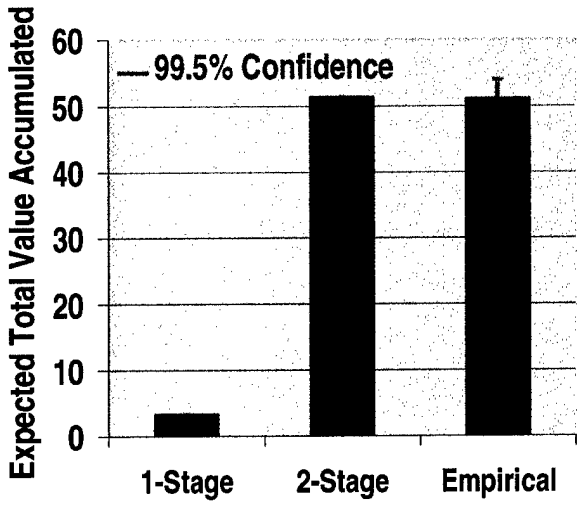


Figure 44 Prediction Accuracy of Different Design Models for the 2-Stage/1-Wave Retasking Problem with Unknown Initial State x_0

the figure that the analytic model that accounts for the retasking loop closure is statistically equivalent to the true expected performance. Furthermore, the analytic model that does not account for the retasking loop closure provides a poor prediction and is only 6% of the true expected performance. Thus, it is seen from this and the previous assessment that 2-stage/1-wave retasking analytic prediction model does provide a good approximation to the true expected performance. Furthermore, not unexpectedly, the 1-Stage/1-Wave analytic prediction model does not provide a good approximation to the true expectation.

Having presented the prediction accuracy, the question remains whether the higher fidelity, i.e. proactive, prediction model that anticipates the future retasking loop closures improves the base mission control solution. To perform this assessment, two mission queues were generated: one using the 2-stage analytic predictor that anticipates the retasking loop closure and one using the 1-stage analytic predictor that neglects the retasking loop closure. In both cases, the Surrogate Method, which was presented in Section 4.3.3, was used to search the control space, and the initial status of the enemy and thus the initial state x_0 is known. Figure 45 illustrates the two mission queues produced.

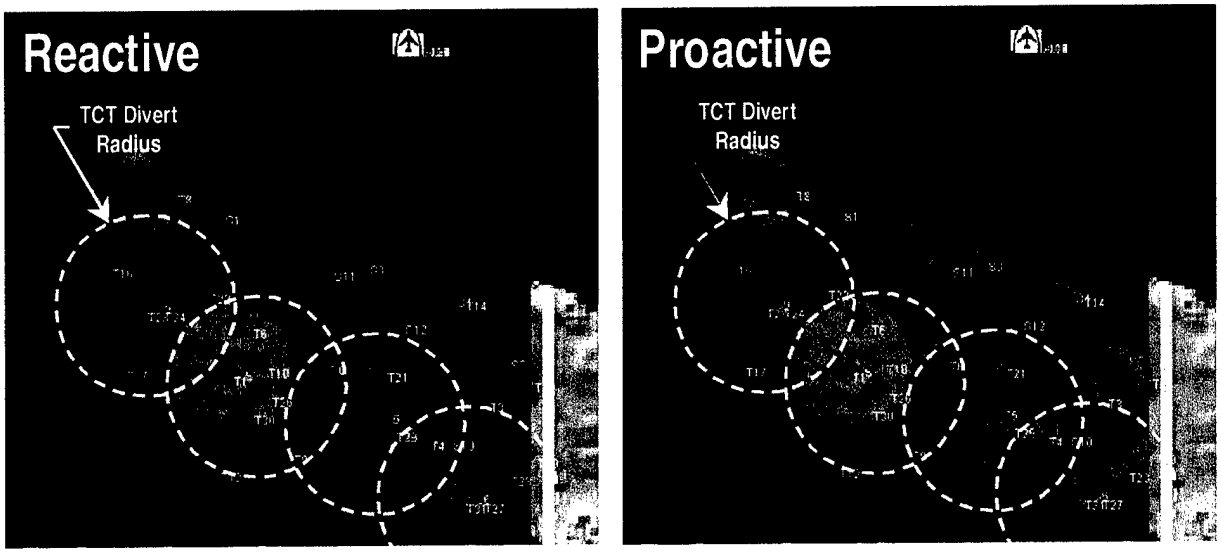


Figure 45 Behavioral Comparison of Proactive Versus Reactive Control Strategy for 2-Stage/1-Wave Retasking Problem

It is seen from this figure that the mission queue generated using the reactive prediction model assigns air packages to TCT locations. In comparison, the mission queue generated using the proactive prediction model does not assign air packages to any TCT locations, but instead strategically locates air packages such that each TCT has a minimum of two retask options. The

significance of this differing mission assignment is that the 2-stage model anticipates the fact that air packages in the vicinity of the TCT regions can be retasked in the event that a TCT emerges; if no TCT emerges, the air packages strike normal targets. Thus, for the 1-stage mission queue, air packages fly to TCT location and if no TCT emerges, the air package cannot achieve any positive value. On the other hand, for the 2-stage solution, air packages fly to normal targets in the vicinity of TCTs. If a TCT emerges during ingress, a retasking solution exists since an air package will be in range; if no TCT emerges, the air packages proceed to their normal targets and achieve positive value. Thus, by anticipating the TCT emerge event and the subsequent loop closure, resources can be more effectively tasked.

Having illustrated the behavioral differences between the base mission solutions, the performance numbers will now be presented. Figure 46 illustrates the performance and empirical performance prediction for the two mission queues. Note, empirical results were obtained using 10,000 Monte Carlo samples. It is seen from this figure that 2-stage control solution exhibits a statistically significant performance improvement over the 1-stage control solution. Furthermore, as expected, the predicted performance of the two solutions is pessimistic. In the case of the 1-stage solution, the predicted performance does not account for any retasks, whereas the empirical solution does permit reactive retasking to a TCT emerge event. On the other hand, the lower predicted performance of the 2-stage model was identified in previous paragraphs to be related to the approximations relative to the uncertain loop closure time.

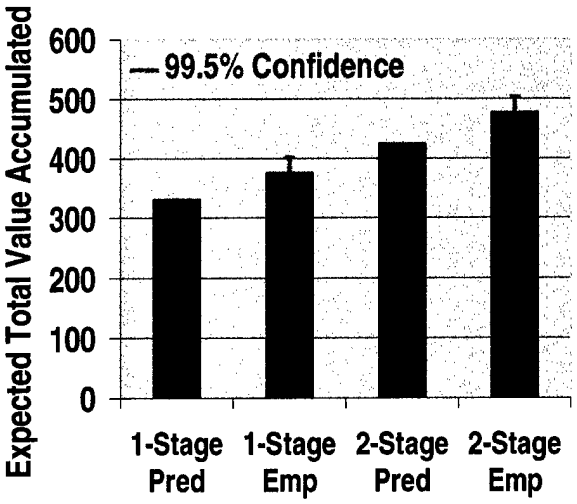


Figure 46 Control Performance for the 2-Stage/1-Wave Retasking Problem

The final piece to the performance story is to compare the proactive versus reactive controller complexity for this particular scenario. Figure 47 illustrates the number of 1-stage prediction function calls required to produce a control solution for the different control approaches. Note, as highlighted in Section 4.6.3, a single stage prediction model may require hundreds of one-stage prediction function calls to produce the estimated performance for a control option. It is seen from this figure that the 2-stage prediction algorithm only requires a marginal 0.5 orders of magnitude more function calls. In terms of clock time, the reactive, 1-stage algorithm computes a base mission queue in ~1 minutes, whereas the proactive, 2-stage algorithm computes a higher quality mission queue in ~5 minutes.

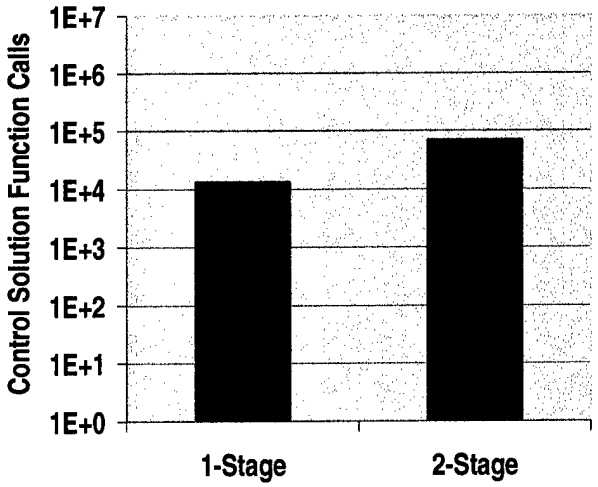


Figure 47 Controller Computational Performance for the 2-Stage/1-Wave Retasking Problem

To summarize these results, the 2-Stage/1-Wave with Retasking problem illustrates the benefits of anticipating future TCT emerge events and subsequent retasking controller loop closures. It was shown that the analytic prediction model that approximates the future loop closure does provide an accurate prediction for a given control option both for the cases where the initial enemy status is deterministic or defined by distributions. Furthermore, it was shown that using the analytic prediction model that anticipates the retasking loop closure produces proactive control solutions that strategically position air packages near TCT regions. Finally, experimental results show that the anticipatory approach provides a statically significant performance improvement over control solution that only reacts to the TCT emerge event.

5.4 2-STAGE/2-WAVE PROBLEM

In the previous section, the benefits of proactive versus reactive control were illustrated for the case when the prediction horizon was 1-wave but included the potential for retasking loop closures. In this assessment, the complexity of the problem is increased by extending the prediction horizon to include the second wave, without the potential for retasking loop closures.

Thus, this experiment will highlight the benefits of performing proactive control over a 2-wave problem. Accordingly, the 2-Stage/2-Wave prediction model presented in Section 4.6.3 will be used for this experiment.

To show this benefit, the scenario was constructed such that all normal targets have terminal time constraints, i.e. expiration deadlines, on their value, and these constraints were established such that they become active during the second wave execution. Additionally, routing risk was added to the control space; in this context, the controller could choose either a high or low risk route for the entire wave. The combination of target value deadlines and routing provides a design trade of managing risk and execution time—through either target tasking or route selection—to maximize the two-wave performance. Thus, for this two-wave problem, the controller determines the initial mission queue and routing option using a two-wave prediction model. Then, the air packages launch at $t=0$ and ingress to their respective targets, deploy their munitions, and egress to base. When all air packages return to base, the controller then determines the second wave mission queue and routing option using a 1-wave prediction model. Again the air packages launch from base, ingress to their respective targets, deploy their munitions, and return to base. The second wave—and the experiment— is complete when all air packages return to base. For this experiment, all mission queues were generated using the Maximum Marginal Return assignment algorithm discussed in Section 4.5.3.

As noted in Section 4.6.3, the difficulty of a 2-wave prediction is modeling the loop closure and subsequent mission generation at the end of the first wave. This is a difficult problem because there is a combinatorially large number of possible states at the end of the first wave. As a further complication, all of the assignment algorithms outlined in Section 4.3 produce a discrete mission queue for a discrete number of resources; thus, there is no control algorithm that maps a surviving aircraft distribution to a mission queue distribution. Given these complexities, a variety of 2-stage/2-wave prediction models will be assessed in this section. As in the previous sections, an assessment of the prediction quality and control performance will be made for each of these approaches.

To perform the prediction accuracy assessment, the two-wave problem was set up where the initial mission queue was defined *a priori*; thus, the only loop closure occurs when the first wave ends and the base controller determines the second wave mission queue. Figure 48 illustrates the 2-stage/2-wave prediction problem where the initial state includes the initial mission queue and the enemy status at $t=0$. The battlespace state is propagated to the 1st wave return to base, and based on some approximation, the 2nd wave mission queue is modeled. Given the 2nd wave mission queue and the distribution over the enemy status, the battlespace state is propagated to the end of the 2nd wave. Based on the distribution of the battlespace at the end of the second wave, the performance metric is computed.

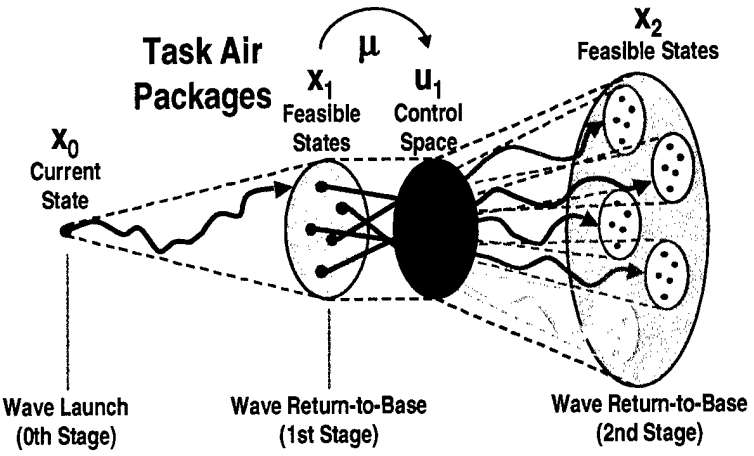


Figure 48 Battlespace State Propagation for 2-Stage/2-Wave Problem

To perform this assessment, the 2-wave prediction models presented in Section 4.6.3, which include random sampling, certainty equivalence, and aggregated approximation, are compared to experimentally generated expected performance. Figure 49 illustrates the prediction accuracy of the different two-wave prediction models compared to empirical evaluation. For the empirical evaluation, 10,000 Monte Carlo evaluations were performed to generate the empirical performance prediction. It is seen from this figure that the

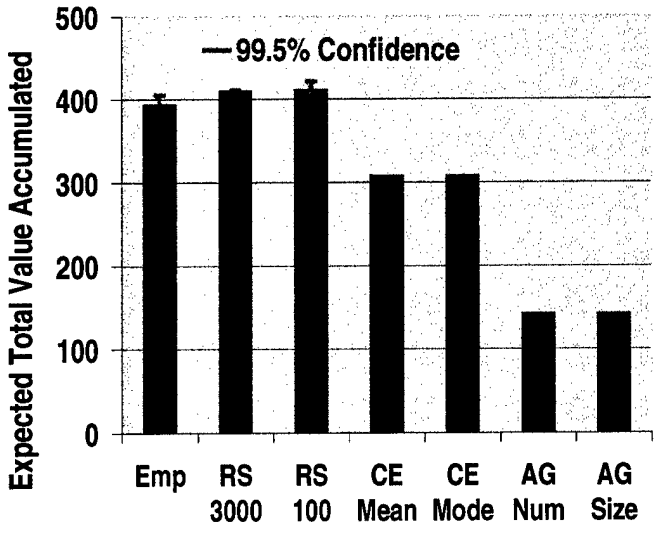


Figure 49 Prediction Accuracy of Different Design Models for the 2-Stage/2-Wave Problem

two random sampling approaches are statistically equivalent to the empirical expected performance. Note, RS # represents random sampling using # realizations of the battlespace state at the end of the first wave; for each realization, the mission controller is called to produce the second wave mission queue. It is also seen from this figure that the two certainty equivalence approaches are within 21% of the true expected performance. Note, CE Mean/Mode represents certainty equivalence using the mean/mode of the battlespace state at the end of the first wave to generate a single 2nd wave mission queue. Finally, it is seen from this figure that the aggregate approximation approaches are within 64% of the true expected performance. Note, AG Num represents the aggregation approach that reduces the number of air packages that are launched and AG Size represents the approach that reduces the size of the air packages launched. Thus, a spectrum of 2-wave prediction models with varying accuracy, randomness, and computation complexity exist. We now direct our attention to how well these models support the proactive control decision being made during the first wave mission generation.

Having presented the prediction accuracy, the question remains whether the higher fidelity prediction models improve the initial base mission control solution. To perform this assessment, initial mission queues were generated using all of the 2-wave prediction models presented above. Again, the MMR assignment algorithm was used to search the control space. To provide a comparison between proactive and reactive control techniques, mission queues were generated using the 1-stage/1-wave prediction model and the 2-stage/2-wave prediction models. In both cases, the initial status of the enemy and thus the initial state x_0 is known. Figure 50 illustrates the initial mission queues produced by the reactive control technique and a representative proactive control technique using one of the 2-stage/2-wave prediction models. It is seen form this figure that the proactive control approach chooses a low risk route for the initial wave but chooses not to attack the furthest distance targets. In contrast, the reactive control solution also chooses low-risk routes, but chooses to strike targets that require longer ingress and egress times. The net impact of further targets and low risk routes increases the 1-wave control solution expected execution time by 30% over that of the two-wave control solution. This 30% increase in expected wave execution time severely limits the target opportunities during the second wave.

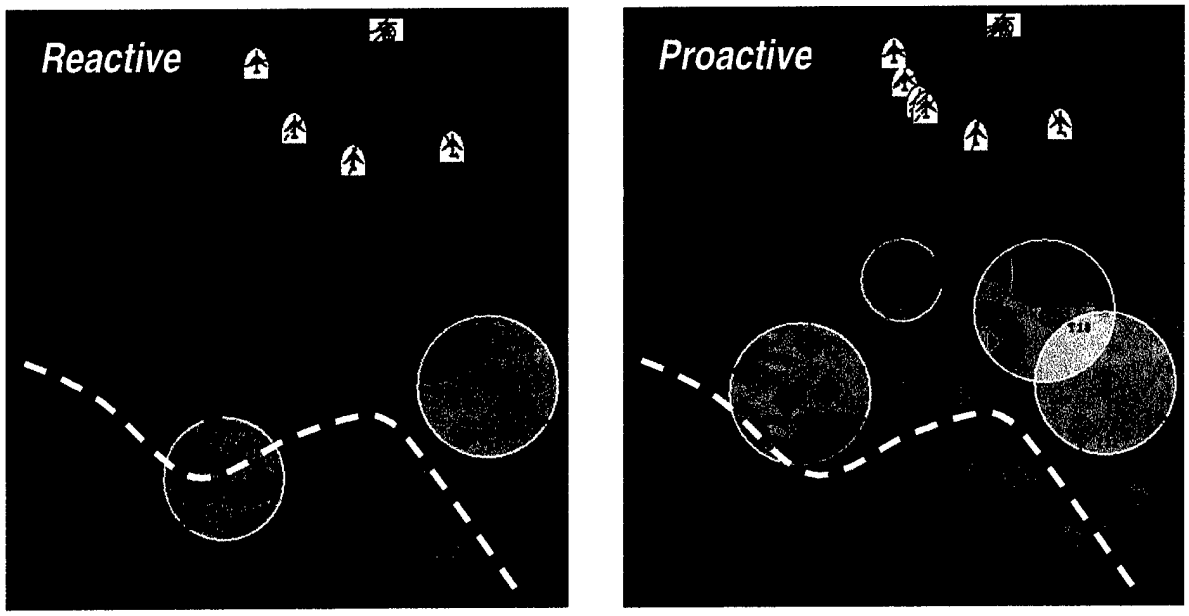


Figure 50 Behavioral Comparison of Proactive Versus Reactive Control Strategy for 2-Stage/2-Wave Problem

Having illustrated the behavioral differences between the base mission solutions, the performance numbers will now be presented. Figure 51 illustrates the empirical performance for the different mission queues generated using 1-stage and 2-stage prediction models. Note, empirical results were obtained using 10,000 Monte Carlo samples. It is seen from this figure that the performance of the 2-stage control solutions is mixed. From the prediction assessment above, it is known that the random sampling approach provides an accurate, albeit noisy, estimate of the expected performance. Thus, this prediction model serves as a baseline to highlight the

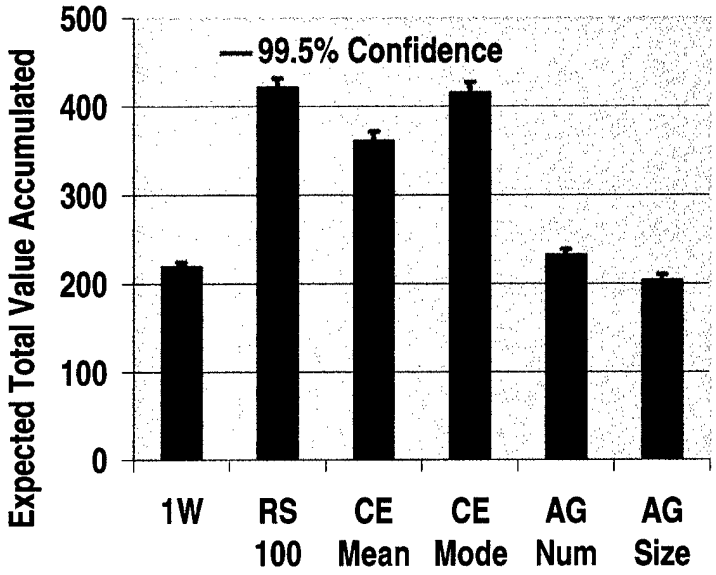


Figure 51 Control Performance for the 2-Stage/2-Wave Problem

achievable performance improvement over the 1-stage approach. Furthermore, the random sampling approach provides a baseline to compare the different 2-wave approaches. It is seen from this figure that the baseline proactive control approach provides a 93% improvement over the expected performance of the reactive control approach. Relative to the other 2-wave approaches, it is seen that the certainty equivalence using the mode is statistically equivalent to the baseline, the certainty equivalence using the mean is within 14% of the baseline, the aggregate approach reducing the number of 1st wave air packages is within 45% of the baseline, and the aggregate approach reducing the size of the 1st wave air packages is within 52% of the baseline. Thus, the certainty equivalent approaches provide superior performance over the aggregate approaches. Furthermore, it is believed that the certainty equivalence using the mode provides superior performance over the certainty equivalence using the mean because the mode amplifies the differences between the good and the bad control options.

The final piece to the performance story of the different proactive, 2-wave approaches is to view the controller complexity for this particular scenario. Figure 52 illustrates the number of one-stage prediction function calls required to produce a control solution for the different control approaches. Note, as highlighted in Section 4.6, a single 2-stage prediction model may require hundreds of one-stage prediction function calls to produce the estimated performance for a control option. It is seen from this figure that the baseline solution requires approximately 5.0 orders of magnitude more function calls than the 1-wave control solution. Relative to the 2-wave control solution, the certainty equivalence approaches require approximtely 2.0 orders of magnitude less function call than the baseline, and the aggregate approaches require approximately 4.5 order of magnitude fewer function calls.

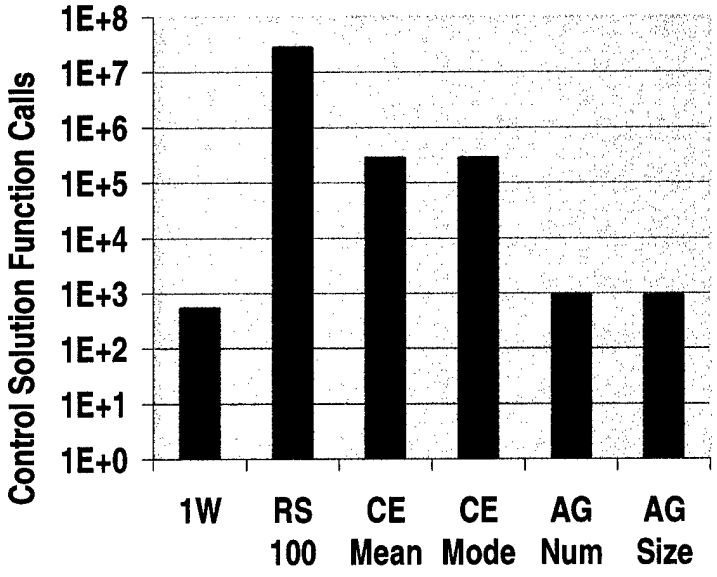


Figure 52 Controller Computational Performance for the 2-Stage/2-Wave Problem

In summary, this assessment highlighted the benefits of performing proactive versus reactive control over a 2-wave problem. As highlighted in Section 4.6.3, there are a variety of ways to approximate the loop closure that generates the 2nd wave mission queue, and we have chosen to illustrate random sampling, certainty equivalence, and aggregate approaches where the random sampling approach was used as a baseline to compare prediction accuracy and control solution quality. From this assessment, it was determined that proactive control, which anticipated the deadlines in the second wave and determined 1st wave missions to minimize risk and execution time, provides a substantial performance improvement over non-anticipatory, i.e. reactive, control strategies. Furthermore, it was shown that the certainty equivalence control approach does provide an accurate prediction of the expected 2-wave performance and does produce high-quality control solutions at a substantial reduction in computation complexity when compared to the baseline. Finally, it was shown that the aggregated approaches neither provide an accurate prediction of the 2-wave predicted performance nor provide quality control solutions.

5.5 2-STAGE/1-WAVE MODEL AOR TASKING UNDER UNCERTAINTY PROBLEM

In the previous section, the benefits of proactive versus reactive control were illustrated for a 2-wave problem with stringent terminal constraints. In this assessment, the complexity of the problem will be increased to include AOR tasking under uncertainty. Thus, this experiment will highlight the benefits of anticipating the arrival of information and closing the loop in the context of AOR tasking. Accordingly, the 2-Stage/1-Wave AOR Tasking prediction model presented in Section 4.6.6 will be used for the experiment.

To show this benefit, the standard scenario presented in Section 5.1 was modified by making all targets of the TCT type and by including AOR points and ISR collection assets. Given that ISR collection is explicitly modeled in this scenario, perfect state information about the enemy status is not assumed for this experiment during execution. Note that in previous experiments, the battlespace state at loop closures was always perfectly observed, and uncertain state information was only captured within the prediction model for future loop closures. Figure 53 illustrates the modified scenario used for this experiment. It is seen in this figure that the scenario now includes two AORs and ISR assets that have *a priori* defined missions to fly from left to right. Since we only have perfect state information when an ISR asset is within range of an enemy asset, our knowledge of the enemy state is represented by a distribution at any given time. This knowledge is represented by the rings in the SAM's lethal range where the different colors represent different modes. Thus, the area of the ring represents the probability of being in a particular state. If an ISR asset is within range of the SAM, the color of the SAM's range is solid to reflect perfect observation. Once the ISR asset is out of range of the SAM site, the status information begins to decay according to a transient Markov process, and eventually achieves a steady state distribution. Figure 54 illustrates this transient behavior.

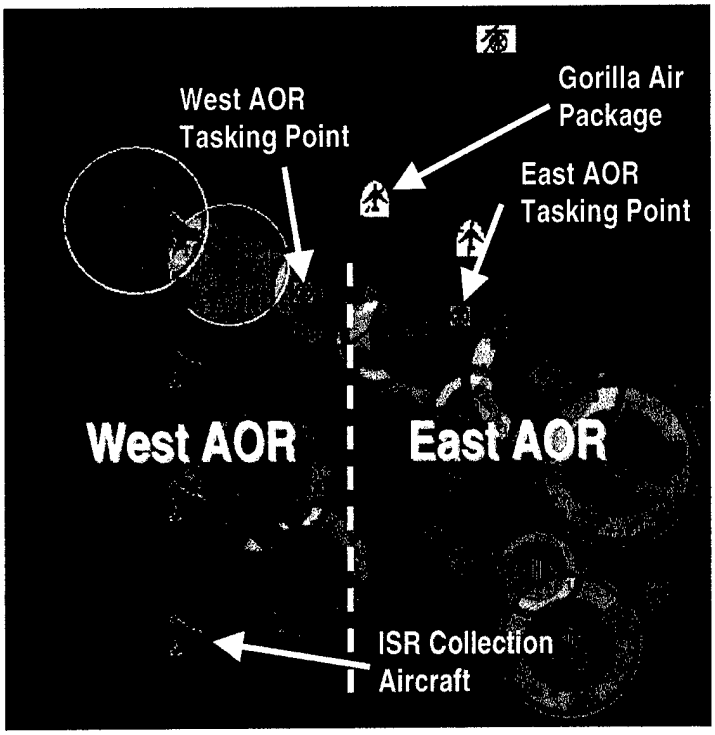


Figure 53 Modified Demonstration Scenario Used for AOR Tasking Under Uncertainty Problem

Having no information about the enemy status at $t=0$, it is reasonable to assume that all assets are in a steady state condition. For this experiment, the steady state distribution for the targets is $P_{known}=0.5$, $P_{unknown}=0.5$, and $P_{dead}=0$. For the SAMs, the steady state distribution is such that $P_{active}=0.4$, $P_{inactive}=0.35$, $P_{unknown}=0.25$, $P_{repair}=0$, and $P_{dead}=0$.

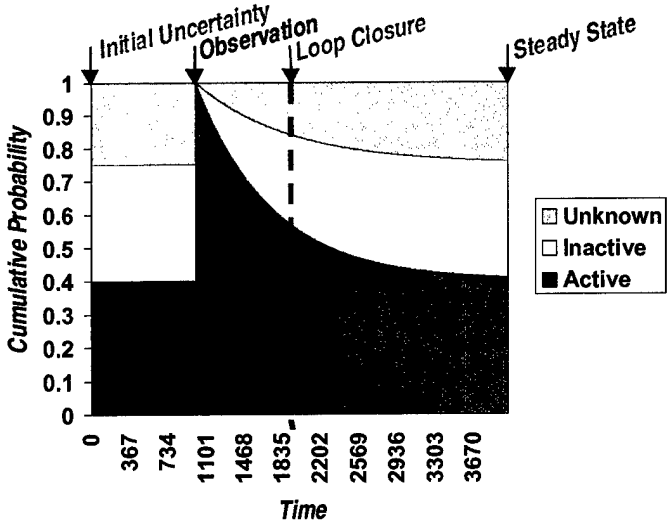


Figure 54 Markov Transient Response of Known SAM Site Status Distribution

The execution sequence for this scenario is as follows:

- Based on uncertain initial information, it is assumed that enemy status at $t=0$, i.e. x_0, is governed by steady state distributions.
- Given this information, the controller composes and tasks gorilla air packages to the AORs.
- The gorilla air packages launch at $t=0$ in ingress to their respective AORs using straight line routes.
- During ingress, the ISR assets along with the gorilla air packages collect observations about the enemy status.
- Upon each gorilla air package's arrival to its AOR, a controller decomposes it into smaller air packages and assigns them to particular target locations based on the information collection.
- Air packages ingress via straight line routes to their respective target locations, deploy munition, and egress back to base.
- Wave ends when all air packages return to base.

Given these modifications to the base scenario and the execution sequence above, the proactive base mission controller must anticipate the ISR collection, information degradation, and the AOR loop closure mapping in order to make optimal gorilla package composition and tasking. Note that for all experimental results presented in this section, the base mission queues, i.e. gorilla air package composition and tasking, are generated using the Surrogate Method discussed in Section 4.5.6. Likewise, all AOR taskings, i.e. decomposition of gorilla air package and air package composition and tasking, are generated using the Maximum Marginal Return assignment algorithm discussed in Section 4.5.3.

As noted in Section 4.6.6, the difficulty of a 2-stage/1-wave AOR tasking under uncertainty prediction model is modeling information arrival and degradation and the loop closure upon arrival to the AOR. This is a difficult problem because there is a combinatorial large number of possible states upon arrival to the AOR due to gorilla package attrition and information arrival and degradation. As a further complication, all of the assignment algorithms outlined in Section 4.3 produce a discrete mission queue for a discrete number of resources; thus, there are no control algorithms that map a surviving aircraft distribution to a mission queue distribution. Given these complexities, a variety of AOR tasking prediction models will be assessed in this section. As in the previous sections, an assessment of the prediction quality and control performance will be made for each of these approaches.

To perform the prediction accuracy assessment, the 1-wave AOR tasking problem was set up where the initial gorilla mission queue was defined *a priori*. Thus, the only loop closures occur when the gorilla air packages arrive at the AORs; upon arrival, the AOR tasking controller determines the low level air package composition and tasking to target locations. Figure 55 illustrates the 2-stage/1-wave AOR tasking under uncertainty prediction problem where the initial state includes the initial mission queue and the enemy status at $t=0$. The battlespace state is propagated to the AOR rendezvous point, and based on some approximation, the AOR tasking is modeled. Given the AOR tasking and the distribution over the enemy status, the battlespace state is propagated to the end of the 1st wave. Based on the distribution of the battlespace at the end of the second wave, the performance metric is computed.

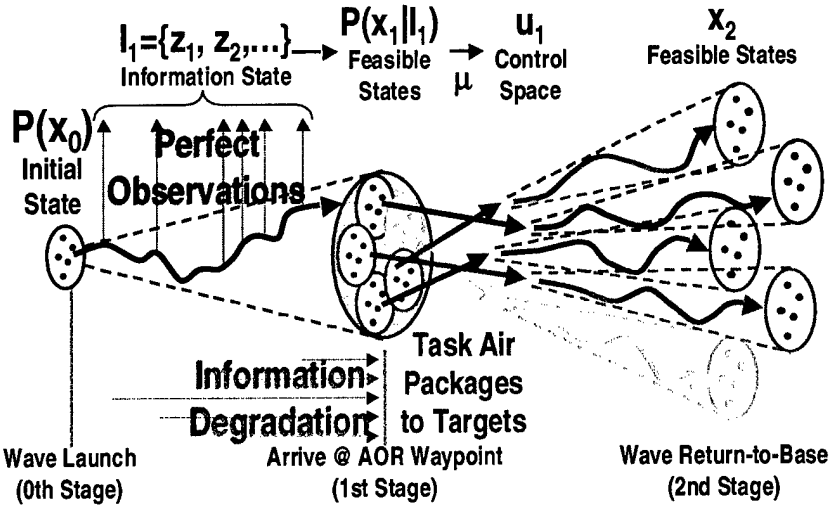


Figure 55 Battlespace State Propagation for 2-Stage/1-Wave AOR Tasking Problem with ISR Collection and Degradation

To perform this assessment, the 2-stage AOR prediction models presented in Section 4.6.6, which include random sampling, certainty equivalence, and partial OLF, are compared to experimentally generated expected performance. As an additional comparison, the 1-stage prediction model that does not anticipate future information arrival and control decisions is included. Figure 56 illustrates the prediction accuracy of the different two-wave, AOR prediction models compared to empirical evaluation.

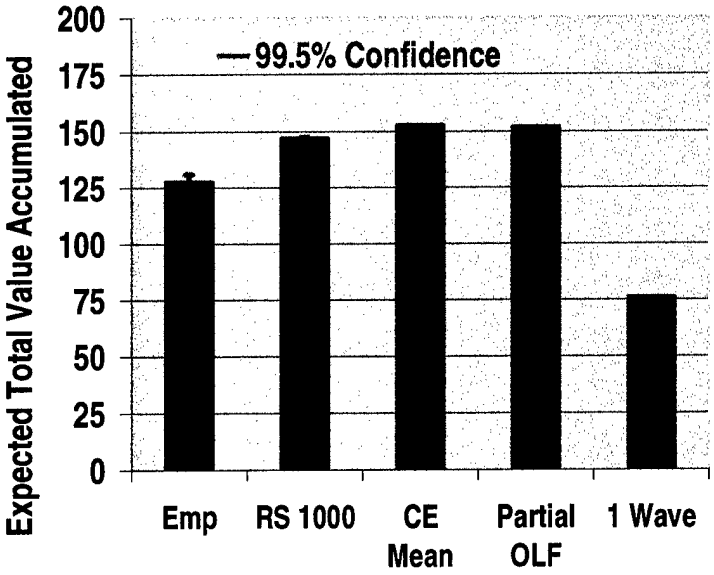


Figure 56 Prediction Accuracy of Different Design Models for the 2-Stage/1-Wave AOR Tasking Under Uncertainty Problem

For the empirical evaluation, 23,500 Monte Carlo evaluations were performed to generate the empirical performance prediction. It is seen from this figure that the 2-stage AOR prediction algorithms are consistently optimistic by approximately 15%. As discussed in the previous section, the random sampling approach should provide an accurate, albeit noisy, estimate of the expected performance. However, as seen in the above figure, the random sampling approach is producing an optimistic estimate of the expected performance. From a thorough evaluation of prediction model, it was determined that there is a model mismatch between the prediction model and the simulator. In the simulator, information degradation begins when the ISR collection asset flies out of range of the enemy asset; however, in the prediction model, information degradation begins immediately after detection, i.e. when the asset first comes into range. Thus, the prediction model is overestimating the transient time for information degradation. Finally, it is seen in the above figure that the 1-stage prediction model, which does not account for information arrival and subsequent AOR loop closures, is only within 40% of the true expected performance.

Having presented the prediction accuracy, the question remains whether the higher fidelity prediction models improve the initial gorilla air package assignment. To perform this

assessment, initial mission queues were generated using all of the 2-stage AOR prediction models presented above. To provide a comparison between proactive and reactive control techniques, mission queues were generated using both the 1-stage/1-wave and 2-stage/1-wave prediction models. Figure 57 illustrates the initial mission queues produced by the proactive and reactive control techniques. It is seen from this figure that the proactive control approach chooses to send all assets to the AOR for which information becomes available. This solution is attained by all the 2-stage controllers, i.e random sampling, certainty equivalent, and partial OLF. On the other hand, the 1-stage controller, not able to recognize the arrival of information, let alone its benefit, selects a uniform allocation of resources to AORs. These results are consistent with our expectations.

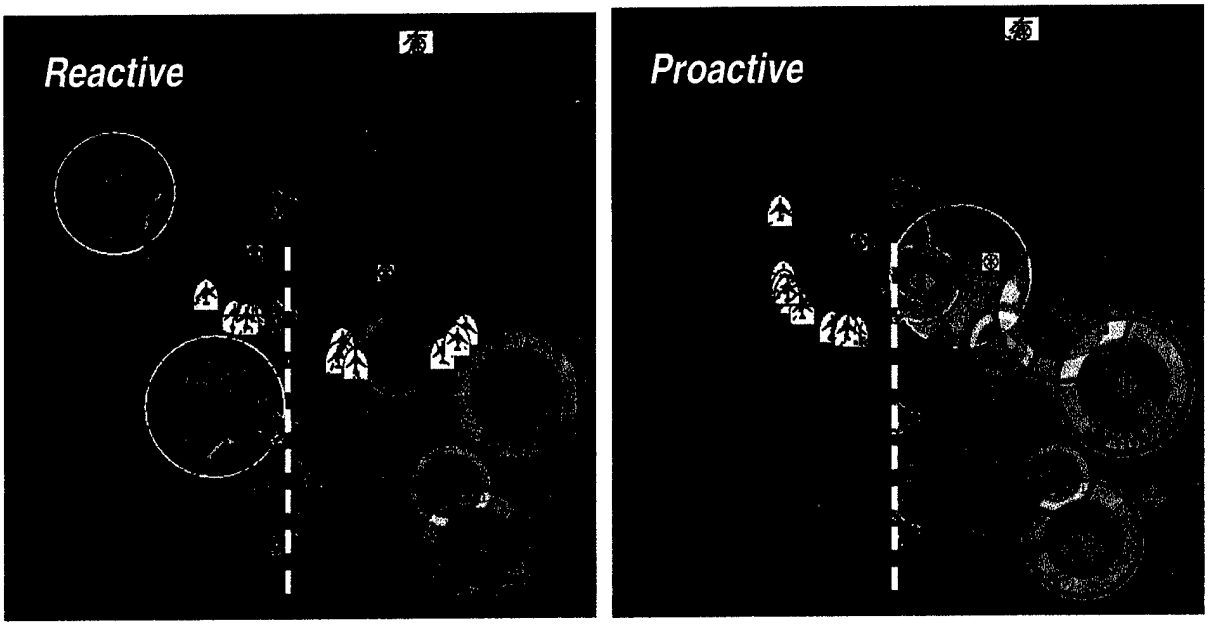


Figure 57 Behavioral Comparison of Proactive Versus Reactive Control Strategy for 2-Stage/1-Wave AOR Tasking Under Uncertainty Problem

Having illustrated the behavioral differences between the base mission solutions, the performance numbers will now be presented. Figure 58 illustrates the empirical performance for the different mission queues generated using 1-stage and 2-stage prediction models. Note, empirical results were obtained using 20,400 Monte Carlo samples. It is seen from this figure that the performance of the 2-stage control solutions are all identical, which is not surprising since there mission queues were identical. In comparison to the reactive control approach, the proactive controllers produced a 106% improvement in expected performance.

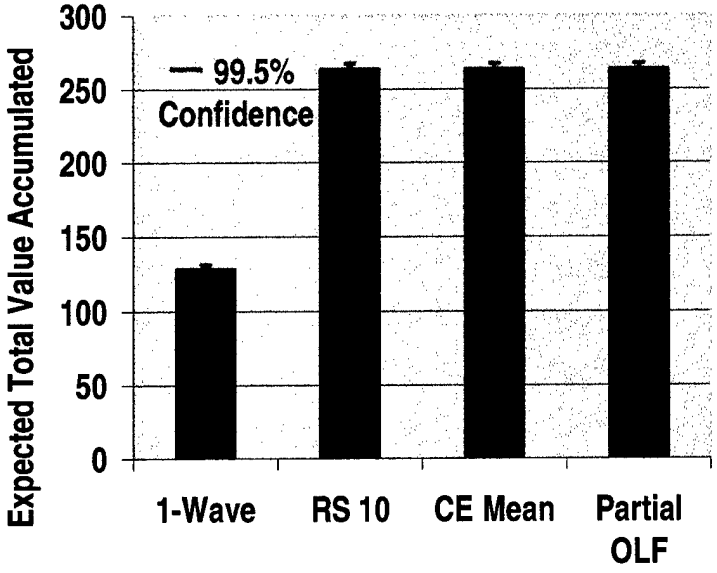


Figure 58 Control Performance for the 2-Stage/1-Wave AOR Tasking Under Uncertainty Problem

The final piece to the performance story of the different proactive, 2-stage approaches is to view the controller complexity for this particular scenario. Figure 59 illustrates the number of one-stage prediction function calls required to produce a control solution for the different control approaches. Note, as highlighted in Section 4.6, a 2-stage prediction model may require hundreds of one-stage prediction function calls to produce the estimated performance for a control option. It is seen from this figure that the baseline solution requires approximately 2.5

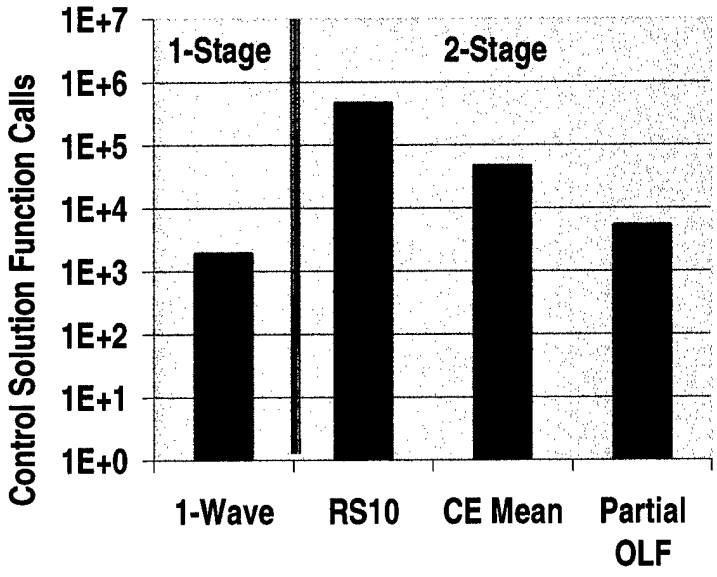


Figure 59 Controller Computational Performance for the 2-Stage/1-Wave AOR Tasking Under Uncertainty Problem

orders of magnitude more function calls than the 1-wave control solution. Relative to the 2-wave control solution, the certainty equivalence approaches require approximtely 1.0 orders of magnitude less function call than the baseline, and the aggregate approaches require approximately 2.0 order of magnitude fewer function calls.

In summary, this assessment highlighted the benefits of performing proactive versus reactive control for a 1-wave AOR tasking under uncertainty. It was shown that control strategies that anticipate future information collection, degradation, and loop closures can provide a substantial performance improvement over control strategies that only react to future information.

6 CONCLUSIONS

This research, performed by ALPHATECH, focused on providing military commanders with the ability to perform real-time dynamic control of military air operations using near optimal mission replannning for a 24-hour segment of a JAO campaign using control algorithms that anticipate possible mission modifications due to uncertain future events. The near real-time, near optimal control decisions being produced consist of the generation/modification of mission definitions for both assets at base and airborne with the performance goal of achieving the specified JFACC objective while minimizing the friendly asset losses. The mission definition includes assignment of resources to targets, high-level routing (by specification of waypoints), strike package composition, weapon composition, and desired time-on-target. The primary benefit of this technology is agile and stable control of distributed and dynamic military operations conducted in inherently uncertain, hostile, and rapidly changing environments.

The JAO problem size investigated includes approximately 100 targets/threats and a mixture of 50 airborne assets taken from two generic airborne asset types: strike and weasel aircraft. Strike aircraft attack targets and weasel aircraft strike surface-to-air threats. Risk to the air packages is introduced via threats such as were surface to air missiles. The size of operation we chose for our study assumes that some form of geographic decomposition of the battle space has been specified, and that we are concerned primarily with achieving the specified missions for a 24 hour period. Hence, there is implicit value in saving assets for future operations beyond the specified horizon.

For the above JAO problem, there are many interesting dynamics that make it challenging. The scarcity of aircraft resources forces multiple "turns" of the aircraft in order to service all of the targets. There are also multiple sources of uncertainty in the problem. There is uncertainty in the status and location of enemy assets. Targets may be known, unknown, hiding, emerging, time critical, or destroyed. Threats may be active, inactive, unknown, repairing, or destroyed. There is also uncertainty in the interaction between enemy and friendly assets that depends on the characteristics of the target/threat and the relative position and composition of the air package, i.e. number and position of strike and weasel.

Given this highly uncertain and rapidly changing environment, the JAO control problem can be viewed as a dynamic decision problem under uncertainty. This class of problems can be formulated as a Markov decision problem. Exact techniques for control design using this

approach, such as Stochastic Dynamic Programming (SDP), are computationally expensive, and do not scale up to the size of the JAO problem of interest. A subtle but significant attribute of the Markov decision problem formulation is that it produces control strategies that anticipate the effects of future contingencies, and evaluates the possible actions over all possible future states, by modeling the future information arrival and control decisions. It is this fact that produces *proactive* versus *reactive* control behaviors. This proactive attribute is desirable for stable and agile control of the JAO enterprise because future information arrival and control opportunities are dependent on stringent spatial, temporal, and coordination constraints.

Given the strengths and weaknesses of the SDP algorithm, this research focused on developing Approximate Dynamics Programming (ADP) strategies that provide the desirable *proactive* control behaviors but with near real-time computation effort. The control design technology is based on combining hybrid state modeling techniques for developing statistical dynamical models relating mission decisions to evolution of objects in the battlespace, together with ADP control design techniques that have demonstrated real-time, proactive performance for other relevant military problems. Accordingly, a spectrum of ADP control techniques were developed for the JAO problem; these techniques were developed in discrete event simulations of JAO scenarios. The major accomplishments of the research were:

- Translated the JAO Control Enterprise into a Dynamical Hybrid State, Discrete Event, Stochastic Decision Making Problem
- Integrated Emerging ADP Technologies into JAO Feedback Controllers
- Experimentally Demonstrated the Benefits of Feedback Control
- Experimentally Demonstrated Benefits of Approximate Optimal Control
- Developed Innovative Hybrid, Multi-Rate Control Architecture
- Developed Computationally Efficient Control Algorithms that Produce Operationally Consistent Behaviors
- Extended Control Algorithms to Accommodate Hierarchical Mission Tasking and ISR Information Collection

In summary, our investigations demonstrated the feasiblility of automating military operations planning to provide real-time, near-optimal control strategies that achieve operational objectives while minimizing asset losses. By adopting a hybrid, multi-rate control architecture, we were able to tailor the application of control at a time scale appropriate to the operational situation at hand. Within the proposed multi-rate architecture, we developed a spectrum of ADP

control strategies that produce a range of control decisions, ranging from immediate restasking or abort decisions to preplanned multiple wave tasking. The solution quality and computation performance of these algorithms was tested and verified in a JAO discrete event simulator. Our experiments show that the ADP strategies were able to produce operationally consistent, proactive control strategies that anticipated likely contingencies and positioned assets for opportunities of recourse all in either real-time or near real-time. Furthermore, a scalability assessment indicated that many of the ADP controllers could provide near real-time performance for scenarios with 250 targets with some modest parallel computation.

The results of this investigation can be extended in several important directions. First, the algorithms developed in this investigation can be extended to include further modeling details of a JAO environment, such as detailed weaponeering, additional platform types and missions. In this manner, the algorithms could then form the basis for a decision aid for an Air Operations Center (AOC). This decision aid would assist operators in rapidly replanning missions in the presence of contingencies, and help to generate robust Air Tasking Orders (ATO). Second, the technology developed in this work can be extended to design robust autonomous controllers for automated vehicles conducting uncertain missions.

7 REFERENCES

[B96] Bertsekas, D. P., Dynamic Programming and Optimal Control, Vols. I-II, Athena Scientific, Belmont, MA 1995.

[B99] Berksekas, D.P., "Rollout Algorithms: An Overview," Proceedings of the 38th Conference on Decision & Control, Phoenix, Arizona, December, 1999.

[BT96] Bertsekas, and Tsitsiklis, J. N, Neuro-Dynamic Programming, Athena Scientific, Belmont, MA 1996.

[BTW97] Bertsekas D.P., Tsitsiklis, J.N., and Wu, C., "Rollout Algorithms for Combinatorial Optimization," Journal of Heuristics, Vol. 3, Num. 3, November, 1997, pp.245-262

[SB98] Sutton, R., and Barto, A. G.,. Reinforcement Learning, MIT Press, Cambridge, MA, 1998.

[BC98] Bertsekas, D. P., and Castañon, D. A., "Rollout Algorithms for Stochastic Scheduling Problems," *Journal of Heurisitcs, Vol. 5*, 1999.

[CaPa99] Cassandras, C. G. and C. Panayiotou, "Concurrent Sample Path Estimation for Discrete Event Systems," to appear in Journal of Discrete Event Dynamic Systems, 1999.

[CSSA89] D. A. Castañon, N. Sandell, W. Stonestreet and M. Athans, Advanced Weapon Target Assignment Algorithms Program: Final Report, ALPHATECH TR-440 on Army Strategic Defense Command Contract DASG60-86-C-0050, July 1989.

[Dai97] Dai, L. and Chen, C. H., "Rates of Convergence of Ordinal Comparison for Dependent Discrete Event Dynamical Systems," Journal on Optimization Theory and Applications, V. 94, 1997.

[Vak91] Vakili, P., "A Standard Clock Technique for Efficient Simulation," Operations Research Letters, Vol. 10, pp. 445-452, 1991.

[Glas91] Glasserman, P., Gradient Estimation Via Perturbation Analysis. Boston, MA: Kluwer, 1991.

[HoCao91] Ho, Y.C., and X.R. Cao, Perturbation Analysis of Discrete Event Dynamic Systems. Kluwer Publishing, 1991.

[Cass93] Cassandras, C.G., Discrete Event Systems: Modeling and Performance Analysis. Boston, Irwin, Inc., 1993.

[CassLaf99] Cassandras, C.G., and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publisher, 1999.

[HoCass97] Ho, Y.C., and Cassandras, C.G., "Perturbation Analysis for Control and Optimization of Queueing Systems: An Overview and the State of the Art", in "Frontiers in Queueing" (J. Dshalalow, Ed.), CRC Press, pp. 395-420, 1997.

[HoCass99] Ho, Y.C., Cassandras, C.G., Chen, C.H., Dai, L., "Ordinal Optimization and Simulation," submitted to special issue on simulation to be published by INFORMS 1999.

[Ro83] Ross, S. M., *Introduction to Stochastic Dynamic Programming*, Academic Press, N.Y., 1983.

[BC99] Bertsekas, D.P., Ca stañon, D. A. and et al, "Adaptive Multi-platform Scheduling in a Risky Environment," DARPA-JFACC Advances in Enterprise Control (AEC) Symposium, 1999.

[BC00] Bertsekas, D. P., Castañon, D. A. and et al, "Dynamic Programming Methods for Adaptive Multi-Platform Scheduling in a Risky Environment," DARPA-JFACC Advances in Enterprise Control (AEC) Symposium, 2000.

[GHZ99] Gong, W., Ho, Y., and Zhai W., "Stochastic Comparison Algorithm for Discrete Optimization with Estimation," *SIAM Journal of Optimization*, Vol. 10, No. 2, pp. 384-404, 1999.

[GoCass00] Gokbayrak, K., and Cassandras, C.C., "An On-Line 'Surrogate Problem' Methodology for Discrete Stochastic Resource Allocation Problems, " to appear in Journal of Optimization Theory and Applications, 2000.

[Bran95] Branicky, M.S., "Studies in Hybrid Systems: Modeling, Analysis, and Control," PhD Dissertation, Laboratory of Information and Decision Systems, LIDS-TH-2304, Massachusetts Institute of Technology, June 1995.

[Patek99] Patek, S. D., D. A. Logan and D. A. Castañon, "Approximate Dynamic Programming for the Solution of Multiplatform Path Planning Problems," *Proceedings IEEE SMC '99 Conference*, Richmond, VA, 11/1999.

8 APPENDICES

As noted in previous sections, some of the results that where document in technical memorandums and conference proceeding are being included to compliment the body of this report.

8.1 TYPES OF CONTROL

See attached

Wohletz, J.M., "Optimal Control Solutions for Stochastic Systems," ALPHATECH TM-572, Burlington, MA, 2000.

8.2 AEC 2000

See attached

Bertsekas, D.P., D.A. Castañon, et.al., "Dynamic Programming Methods for Adaptive Multi-Platform Scheduling in a Risky Environment," *DARPA AEC Symposium*, , Minneapolis, MN, July 10-11, 2000.

8.3 AEC 1999

See attached

Bertsekas, D.P., Ca stañon, D. A. and et al, "Adaptive Multi-platform Scheduling in a Risky Environment," DARPA-JFACC Advances in Enterprise Control (AEC) Symposium, 1999.

8.4 ACC 2001

See attached

Wohletz, J.M., D.A. Castañon, and M.L. Curry, "Closed-Loop Control for Joint Air Operations," *IEEE American Control Conference*, ACC01-INV3501, Arlington, VA, 2001.

8.5 SPIE 2001

See attached

Cassandras, C.G., K. Gokbayrak, D. Castañon, J. Wohletz, M. Curry, and M. Gates, "Modeling and Agile Control for a Joint Air Operation Environment," *Proceedings of SPIE 15th Annual Intl. Symposium*, April 2001.

TM-572
Optimal Control Solutions for Stochastic Systems

Jerry M. Wohletz

Abstract

This technical memorandum details the distinction between the different types of control philosophies that are applicable to stochastic optimal control problems. Stochastic optimal control is defined as the determination of control variables or parameters that minimize some well-defined criterion subject to the evolution of a stochastic system [1]. For this control problem, different optimal control policies result from varying the assumptions pertaining to the information state and from varying the assumptions pertaining to the optimal control structure. This paper focuses on distinguishing the differences between control approaches that appear to be equivalent, but in fact produce different behaviors. Given the different optimal control philosophies, a few control techniques will be explored.

Keywords

Stochastic optimal control, approximate dynamic programming, model predictive control.

I. Basic Problem

CONSIDER the a stochastic discrete-time dynamical system [2]

$$x_{k+1} = f_k\left(x_k, u_k, w_k\right), \quad k = 0, 1, ..., N-1 \tag{1}$$

where the state $x_k \in S_k$, the control $u_k \in C_k$, the random disturbance $w_k \in D_k$, and N is the horizon of interest. The control u_k is constrained to take values in a given nonempty subset $U_k(x_k) \subset C_k$ that depends on the current state x_k, i.e. $u_k \in U_k(x_k)\ \forall\ x_k \in S_k$. The random disturbance w_k is characterized by a probability distribution $P_k(\cdot|x_k, u_k)$ that may depend explicitly on x_k and u_k but not on values of prior disturbances $w_{k-1}, \ldots, w_0$.

For all optimization problems, some well-define performance metric is required. Here, we use an additive performance metric in the sense that the cost incurred at time k, denoted by $g_k(x_k, u_k, w_k)$, accumulates over time. Furthermore, we add the possibility of a terminal cost denoted by $g_N(x_N)$. Because of the presence of the random disturbance w_k, the performance metric is a random variable and the optimization must be performed on the expected cost

$$J(x_0) = E_{\substack{w_k \\ k=0,1,\ldots,N-1}} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\} \tag{2}$$

where the expectation is taken with respect to the random disturbances $w_k, k = 0, 1, \ldots, N-1$.

II. Optimal Control Formulations

In the basic problem formulation above, there is no mention of the form of the optimal control solution or the availability of future information; depending on the availability of future information and the structure of the control, fundamentally different optimization problems result. Based on the desired optimal behavior, the distinction between these different formulations is particularly relevant when selecting a control design technique.

There are three different stochastic optimal control formulations presented in the following subsections. The assumptions pertaining to the future information availability and structure of the control is presented along with comments pertaining to implementation. The purpose here is not to identify solution techniques to the optimization problem, but instead gain some understanding between the different optimization formulations.

J. M. Wohletz is a Senior Systems Engineer at ALPHATECH Inc., 50 Mall road, Burlington, MA, 01803 USA, (e-mail: jwohletz@alphatech.com)

A. Open-Loop Optimal Control

The Open-Loop Optimal Control (OLOC) formulation assumes that no future realizations of the state are measurable; as a result, the control structure is purely a function of time and consists of a sequence of actions, i.e. $\mathbf{u_0} = \{\mathbf{u_0}, \mathbf{u_1}, \ldots, \mathbf{u_{N-1}}\}$. Given this assumption, the optimization of (2) subject to (1) over a sequence of controls $\mathbf{u_0} = \{\mathbf{u_0}, \mathbf{u_1}, \ldots, \mathbf{u_{N-1}}\}$ based on $x(0)$ is as follows:

$$\mathbf{u_o^*} = \arg \min_{\mathbf{u_0 \in U}} \mathbf{E}_{\substack{\mathbf{w_k} \\ \mathbf{k=0,1,\ldots,N-1}}} \left\{ \mathbf{g_N(x_N)} + \sum_{\mathbf{k=0}}^{\mathbf{N-1}} \mathbf{g_k(x_k, u_k, w_k)} \right\} \tag{3}$$

subject to the constraints

$$x_{k+1} = f_k\left(x_k, u_k, w_k\right), \quad u_k \in U_k(x_k) \; \forall \; x_k \in S_k, \quad k = 0, 1, ..., N-1 \tag{4}$$

Thus, this optimal control approach produces a control solution that is solely a function of time, and this solution is not updated as more information becomes available.

It is important to note that the optimal control sequence $\mathbf{u_0}$ accounts for the uncertainty $w_0, w_1, \ldots, w_{N-1}$. If the initial state x_0 is uncertain, then the expectation in (3) would be with respect to $x_0, w_0, w_1, \ldots, w_{N-1}$. Also, for large-scale stochastic problem, this optimal solution is by no means trival since the expectation over all disturbances is required.

B. Closed-Loop Feedback Optimal Control

The Closed-Loop Feedback Optimal Control[1] (CLFOC) assumes that future realizations of the state are measurable and that the control structure consists of an optimal rule $\mu_k(x_k)$ — control law — at each time k that maps all feasible realizations of $x(k)$ to feasible $u(k)$, i.e. $u_k = \mu_k(x_k) \in U_k(x_k) \; \forall \; x_k \in S_k$. A sequence of control laws corresponding to each time k forms a policy $\pi = \{u_0, \mu_1(x_1), \ldots, \mu_{N-1}(x_{N-1})\}$. It is important to note the differences between the control structure here and that of OLOC; a control sequence $\mathbf{u_0}$ consists of a set of *'actions'* where a control policy π consists of a set of *'strategies'*. The optimal closed-loop feedback policy π^* for some initial state x_0 is obtained as follows:

$$\pi^* = \arg \min_{\pi \in \Pi} E_{\substack{w_k \\ k=0,1,\ldots,N-1}} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\} \tag{5}$$

subject to the constraints

$$x_{k+1} = f_k\left(x_k, \mu_k(x_k), w_k\right), \quad \pi \in \Pi, \quad k = 0, 1, ..., N-1$$

where Π is the set of all admissible policies such that $u_k = \mu_k(x_k) \in U_k(x_k) \; \forall \; x_k \in S_k$. If the initial state x_0 is uncertain, then the expectation would be taken with respect to $x_0, w_0, w_1, \ldots, w_{N-1}$and the resulting policy would be $\pi = \{\mu_0(x_0), \mu_1(x_1), \ldots, \mu_{N-1}(x_{N-1})\}$.

An important distinction between the CLFOC and OLOC formulation is that the CLFOC problem explicitly accounts for the feedback mechanism in the mathematical formulation by modeling future information arrival and selecting optimal control decisions which depend on the future information. Thus, the solution of the CLFOC problem provides an optimal mapping for all feasible future realizations of the state to control decisions. In comparison, the solution of the OLOC problem produces a sequence of optimal control decisions that are implemented regardless of the future realization of the state.

[1]Dreyfus [1] refers to this approach as Feedback Optimal Control (FOC)

C. Open-Loop Optimal Feedback Control

Open-Loop Optimal Feedback Control (OLOFC) is a term originally coined by Dreyfus [1], and as the name implies, is similar to OLOC in its mathematical formulation but differs in implementation. Like the OLOC formulation, the OLOFC formulation assumes that no future realizations of the state will be measured. As a result, the control structure is purely a function of time and consist of a sequence of actions, i.e. $\mathbf{u_0} = \{\mathbf{u_0}, \mathbf{u_1}, \ldots, \mathbf{u_{N-1}}\}$. However, unlike OLOC, only the initial optimal control $\mathbf{u_0^*(0)} = \mathbf{u_0^*}$ is applied, and the optimization problem is resolved for $\mathbf{u_1^*}$ at time $k = 1$ based on the actual versus expected realization of the state $x(1)$. Then, optimal control $\mathbf{u_1^*(0)} = \mathbf{u_1^*}$ is applied, and the control process repeats until the time $k = N$. Note, as time progresses, the optimization horizon shrinks, i.e. at time n, the planning horizon is $k = n, n+1, \ldots, N-1$.

Since the arrival of future information is withheld from the optimization problem, the optimization is identical to the OLOC formulation; the optimization of (2) subject to (1) over a sequence of controls $\mathbf{u_0} = \{\mathbf{u_0}, \mathbf{u_1}, \ldots, \mathbf{u_{N-1}}\}$ based on x_0 is as follows:

$$\mathbf{u_o^*} = \arg \min_{\mathbf{u_0 \in U}} \mathbf{E}_{\mathbf{k=0,1,\ldots,N-1}}^{\mathbf{w_k}} \left\{ \mathbf{g_N(x_N)} + \sum_{\mathbf{k=0}}^{\mathbf{N-1}} \mathbf{g_k(x_k, u_k, w_k)} \right\} \tag{6}$$

subject to the constraints

$$x_{k+1} = f_k\left(x_k, u_k, w_k\right), \quad u_k \in U_k(x_k) \; \forall \; x_k \in S_k, \quad k = 0, 1, ..., N-1 \tag{7}$$

Again, this optimal control approach produces a control solution that is solely a function of time.

Having stated the similarities and differences between OLOFC and OLOC, we will now focus on the similarities and differences between OLOFC and CLFOC. In both approaches feedback is used in implementation; however, CLFOC explicitly models the feedback mechanism in its mathematical formulation, whereas OLOFC neglects the feedback mechanism in its mathematical formulation. By neglecting the feedback mechanism, the OLOFC solution is simpler to solve. However, what is gained in reducing complexity may be lost in the quality of the solution. By neglecting the feedback mechanism, i.e. future information arrival and subsequent control decision, the anticipatory nature of CLFOC is lost in the OLOFC formulation; as a result, the OLOFC is reactive versus proactive to future realizations of the state. This attribute can have significant impact for systems in which there are significant time delays between future information arrival. To illustrate these differences, Dreyfus [1] presents a solutions to a stochastic optimization problem using the above three formulations.

III. Control Techniques

Given the stochastic nature of the plant in (1), the CLFOC solution will achieve the best performance. If the plant is deterministic, all three optimization formulations would produce the same performance, since feedback is only required if there is uncertainty. A control technique that solves the CLFOC problem is Stochastic Dynamic Programming (SDP). However, this technique is only tractable for small problems. In general, one must resort to approximation techniques. In this section, two different approximate optimal control techniques — in the context of the above formulations — will be presented. The first technique, Rollout, is an approximation to CLFOC, and the second technique, Model Predictive Control, is an approximation to OLOFC. For both techniques, the implication of the approximations on the solution is discussed.

A. Rollout

The Rollout Algorithm (RA) [3], [4] is an approximate CLFOC approach that determines a suboptimal policy on-line based on the actual realization of the state $x(k)$ at time k. Like CLFOC, the RA assumes that future realizations of the state are measurable; however the control strategies to be

used in selecting future decisions are known, consisting of an suboptimal rule $\bar{\mu}(x_k)$ at each future time k that maps all feasible realizations of $x(k)$ to feasible $u(k)$, i.e. $u_k = \bar{\mu}(x_k) \in U_k(x_k)\ \forall\ x_k \in S_k$. As a result, the RA policy has the following form: $\pi_0^{RA} = \{u_0, \bar{\mu}(x_1), \ldots, \bar{\mu}(x_{N-1})\}$. Like OLOFC, only the initial optimal control $\pi_0^{RA^*}(0) = u_0^*$ is applied, and the optimization problem is resolved for $\pi_1^{RA} = \{u_1, \bar{\mu}(x_2), \ldots, \bar{\mu}(x_{N-1})\}$ at time $k = 1$ based on the actual realization of the state x_1. Then, the optimal control $\pi_1^{RA^*}(0) = u_1^*$ is applied, and the control process repeats until the time $k = N$. As before, the optimization horizon shrinks as time progresses, i.e. at time n, the planning horizon is $k = n, n+1, \ldots, N-1$.

Since the feedback mechanism — future information arrival and control decisions — is explicitly modeled in this formulation, the optimization problem is identical to the CLFOC with the exception that optimization over future strategies π is replaced with the known strategy π_k^{RA}, i.e. $\mu_k(x_k)$ is approximated by a known suboptimal baseline strategy $\bar{\mu}(x_k)$. Given that the baseline strategy is known, the optimization over all admissible π_k^{RA} is equivalent to selecting u_0; thus, based on the initial condition x_0, the RA optimization problem is as follows:

$$u_0^* = \arg \min_{u_0 \in U_0(x_0)} E_{\substack{w_k \\ k=0,1,\ldots,N-1}} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \bar{\mu}(x_k), w_k) \right\} \tag{8}$$

subject to the constraints

$$x_1 = f_0(x_0, u_0, w_0), u_0 \in U_0(x_0)$$
$$x_{k+1} = f_k\left(x_k, \bar{\mu}(x_k), w_k\right), \bar{\mu}(x_k) = u_k \in U_k(x_k)\ \forall\ x_k \in S_k, k = 1, ..., N-1$$

The RA is related to the CLFOC formulation since it accounts for future information arrival and control decisions. As a result, RA based controllers will exhibit anticipatory behavior. However, the RA is not as ambitious as the CLFOC formulation, and only provides modest guarantees of near-optimality [4].

Implementation RA requires computing the expectation with respect to random disturbances $w_0, w_1, \ldots, w_{N-1}$, by modeling future information arrival and the resulting control decisions via the baseline strategy. An approach for computing this expectation at a given state x_k and time k is to use Monte Carlo simulations. To implement this approach, we consider all possible controls $u_k \in U_k(x_k)$ and generate a "large" number of simulation trajectories starting from x_k. Thus, the simulated trajectory has the form:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, w_k) \\ x_{i+1} &= f(x_i, \bar{\mu}(x_i), w_i) \quad i = k+1, ..., N-1 \end{aligned} \tag{9}$$

B. Model Predictive Control

The Model Predictive Control (MPC) [5] or Receding Horizon Control (RHC) is an approximate OLOFC technique that solves a deterministic optimization problem. Given that this technical memorandum focuses on stochastic optimal control, the MPC technique requires an approximation to the expectation cost function in (6) that is parameterized by the control sequence $\mathbf{u_k}$. One common approach is to assume that a certainty equivalence principle holds and that the random disturbances $w_0, w_1, \ldots, w_{N-1}$ in (1) can be replaced by their expected values $\bar{w}_0, \bar{w}_1, \ldots, \bar{w}_{N-1}$. Of course, an underlying assumption here is that the random disturbances w_k do not depend on the state x_k and the control u_k. Other approaches can be used to approximate the expectation cost function in (6).

Like the OLOFC formulation, the MPC formulation assumes that no future realizations of the state will be measured. As a result, the control structure is purely a function of time and consist of a sequence of actions, i.e. $\mathbf{u_0} = \{\mathbf{u_0}, \mathbf{u_1}, \ldots, \mathbf{u_{N-1}}\}$. Additionally, the current sequence of optimal control actions $\mathbf{u_k^*}$ are determined on-line at each time k using the current state of the plant x_k and only the first

control in this sequence $\mathbf{u_k^*(0) = u_k^*}$ is applied to the plant. The MPC optimization problem where it is assumed that certainty equivalence principle holds is as follows:

$$\mathbf{u_0^* = \arg \min_{u_0 \in U} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, \bar{w}_k) \right\}} \tag{10}$$

subject to the constraints

$$x_{k+1} = f_k\left(x_k, u_k, \bar{w}_k\right), \quad u_k \in U_k(x_k) \; \forall \; x_k \in S_k, \quad k = 0, 1, ..., N-1 \tag{11}$$

A few notes about MPC are warranted. First, because MPC is based on a OLOFC formulation and does not model the feedback mechanism, the MPC solution can only react to future realizations of the state. As Mayne states [5], "A defect of model predictive control of uncertain systems, not yet widely appreciated, is the open-loop nature of the optimal control problem." Secondly, modeling a stochastic system as deterministic raises the questions of robustness. i.e. the maintenance of certain properties such as stability and performance in the presence of the uncertainty. The inherent robustness properties of the MPC formulation have been studied for systems that are linear with respect to input u and that only have terminal state constraints. In an attempt to add uncertainty to the mathematical formulation, an Open-Loop Min-Max Model Predictive Control formulation has been proposed [5]. However, the solutions are very conservative due to the open-loop nature of the optimization. As Mayne states [5], "... the scenario generated in solving {the open-loop min-max model predictive control problem } does not model accurately the uncertain control problem because it ignores feedback by searching over open-loop control sequences { $\mathbf{u_k}$ } in minimizing the { the cost function }." Because of this deficiency, Mayne recommends a Feedback Mini-Max Model Predictive Control formulation that replaces the control sequence $\mathbf{u_k}$ with a control policy π_k similar to the ones presented above. As Mayne notes, "The feedback version of the model predictive control appears attractive but prohibitively complex."

IV. Conclusions

This technical memorandum presented three different types of control philosophies that are applicable to stochastic optimal control problems. The different optimal control formulations result from varying the assumptions on the future information availability and on the optimal control structure. The Open-Loop Optimal and Open-Loop Optimal Feedback Control solution assume that the no future realizations of the state are measurable in the mathematical optimization formulation. The distinction between the two optimization solutions resides in their implementations. In the Open-Loop Optimal Control formulation, the optimal control sequence generated at time $t = 0$ is not updated. In the Open-Loop Optimal Feedback Control formulation, only the initial control in the optimal control sequence is implemented and the open-loop optimization problem is resolved at the next realization of the state. In contrast to these two approaches, the Closed-Loop Feedback Optimal Control formulation explicitly models the feedback mechanism, i.e. the dependence of future control decisions on future information arrival, in the mathematical formulation. The fundamental distinctions between the open-loop optimizations and closed-loop optimization were that open-loop optimizations produce a set of "*actions*" where as the closed-loop feedback optimization produces a set of "*strategies*". By including the feedback mechanism in the mathematical formulation, the closed-loop feedback optimization is *proactive* versus *reactive* and is able to anticipate likely contingencies; this trait produces guaranteed superior performance in stochastic scenarios.

In addition to making the distinctions between different control philosophies, two approximate stochastic optimal control techniques were discussed. The Rollout Algorithm approximates the Closed-Loop Feedback Optimal Control formulation, and has a similar implementation to the Open-Loop

Optimization Feedback Control. The key approximation in the Rollout Algorithm is that future decisions in the feedback mechanism are modeled by a known suboptimal rule. Next, the Model Predictive Control technique approximates the Open-Loop Optimal Feedback Control formulation. The key point here is that the Open-Loop Optimal expectation cost function is approximated so that a deterministic optimization problem is solved.

ACKNOWLEDGMENTS

This research was supported by the Defense Advanced Research Projects, Information Systems Office (DARPA/ISO) and the Air Force Research Laboratory, Information Directorate, System Concepts & Applications Branch (AFRL/IFSA) under contract number F30602-99-C-0203.

REFERENCES

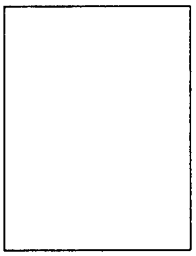
[1] Dreyfus, S. E., "Some Types of Optimal Control of Stochastic Systems," *Journal SIAM Control*, Series A, Vol. 2, No. 1, pp. 120-134, 1962.

[2] Bertsekas, D. P., *Dynamics Programming and Optimal Control, Vol I-II*, Athena Scientific, Belmont, MA, 1995.

[3] Bertsekas, D.P., 'Rollout Algorithms: An Overview' *Proceedings of the 38th Conference on Decision & Control*, Phoenix, Arizona, December, 1999.

[4] Bertsekas, D.P., Tsitsiklis, J.N., and Wu, C., 'Rollout Algorithms for Combinatorial Optimization' *Journal of Heuristics*, Vol. 3., Num. 3, November, 1997, pp. 245-262.

[5] Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O.M., 'Constrained Model Predictive Control: Stability and Optimality' *Automatica*, Vol. 36, 2000, pp. 789-814.



Jerry M. Wohletz received a B.S. degree in aerospace engineering with highest distinctions from the University of Kansas, Lawrence, KS, in 1994, and a M.Eng. and Ph.D. degrees in aeronautics and astronauts from the Massachusetts Institute of Technology, Cambridge, MA, in 1997 and 2000, respectively. He has been with ALPHATECH, Inc., Burlington, MA, since 2000, where he is a Senior Research Engineer. He is currently a Program Leader in the BMC3 business area, and his research interest includes control of large-scale uncertain systems and adaptive control.

Dynamic Programming Methods for Adaptive Multi-platform Scheduling in a Risky Environment[1]

Dimitri P. Bertsekas
Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, Mass., 02139

David A. Castañon
Dept. of Electrical and Computer Engineering, Boston University, Boston, Mass., 02215

Michael L. Curry, David Logan, Cynara Wu[2]
ALPHATECH, Inc., 50 Mall Road, Burlington, MA 01803

Abstract

In this paper, we investigate alternatives to simulation-based approximate dynamic programming methods for adaptive multi-platform scheduling in a risky environment. In a recent effort, we considered rollout algorithms, in which on-line simulation was found to be more reliable than off-line training. Unfortunately, a large amount of computational resources was required to run even a modest number of Monte Carlo simulations. In this paper, we consider alternatives to using simulation. The first approach consists of using limited lookahead policies, which reduce computational requirements by considering value explicitly over a limited horizon and approximating the value of the remaining stages. The second approach decomposes the problem into sub-problems corresponding to platforms. In our computational experiments, we found that many of the variations of these approaches required significantly less computation time than rollout algorithms and also obtained results that were substantially superior.

1. Introduction

The planning and execution of multiple missions in the presence of risk is a problem that arises in many important military contexts. In data collection applications, multiple UAV platforms may be tasked to interrogate different areas, with the risk of platform destruction as each platform pursues its collection mission. In attack air operations, multiple platforms follow risky trajectories to attack enemy targets. For both applications, sensors and communication equipment can provide up-to-date information concerning individual mission and platform status, and thus provide notification of platform losses. This creates opportunities for replanning, using feedback to retask surviving platforms in order to best achieve mission objectives.

In mathematical terms, the above class of problems can be formulated as Markov decision processes. At each stage of the process, decisions are made that affect the evolution of a system state, which is also influenced by random discrete events. The goal is to select the current decision as a function of the current state in order to optimize mission performance.

The principal approach for solving Markov decision problems is dynamic programming (DP). In comparing the available controls at a given state i, DP considers the current stage value, but also takes into account the desirability of the next state j. It "ranks" different states j by using, in addition to the current stage value, the optimal value (over all remaining stages) starting from j. This optimal value is denoted $J^*(j)$ and referred to as the optimal *value-to-go* of j. Unfortunately, it is well known that the computation of J^* is overwhelming for many important problems.

There has been a great deal of research on DP methods that replace the optimal value-to-go $J^*(j)$ with a suitable approximation for the purpose of comparing the available controls at each state. These methods are collectively known as neuro-dynamic programming (NDP). Previously, we applied a particular class of NDP

[1] This work was supported by the Air Force Office of Scientific Research under contract #F49620-98-C-0023.
[2] *Corresponding Author: phone (781)273-3388, fax (781)273-9345, e-mail cynara.wu@alphatech.com*

algorithms, known as rollout algorithms, to risky multi-platform planning and scheduling problems. Rollout algorithms are a form of NDP that exploit knowledge of suboptimal heuristic decision rules to obtain approximations to the optimal value-to-go. We developed several rollout algorithms for risky multi-platform scheduling, using on-line Monte Carlo simulations to evaluate the reference base heuristic policies, and found that they performed significantly better than the base policies as well as off-line training methods. However, even using a modest number of Monte Carlo simulations resulted in large computation times.

In this paper, we consider alternatives to using on-line simulations. In particular, we consider two approaches that use analytic approximations of the value function. We first consider a class of approximation techniques in which the control exercised at a state i is determined by considering the costs accumulated over several stages, and then applying an approximation to the value-to-go from the resulting states. The rollout algorithms considered in our previous effort are a special case in which a single-stage policy is employed and on-line simulation is used in combination with a base heuristic to approximate the value-to-go.

Our second approach involves exploiting the structure of the problem and decomposing the problem into sub-problems, each of which is associated with a corresponding platform. Each sub-problem is solved independently but takes into account the results of previously solved sub-problems.

The paper is organized as follows. In Section 2, we describe the data collection problem which we are addressing. In Section 3, we present the framework for limited lookahead policies. In Section 4, we describe our decomposition approach to the problem. In Section 5, we present some computational results.

2. Example Data Collection Problem

The graph in Figure 1 is an example corresponding to a data collection problem. Each node represents a geographical area of interest with a one-time value (i.e., data may only be collected once from each location). The arcs represent connectivity among the geographical regions and may be successfully traversed with a known probability. Platforms traverse the graph and collect data (value) at each node, or else they are destroyed while traversing specific arcs. If a platform is destroyed on an arc, the value of the destination node is not collected, which can result in retasking other platforms.

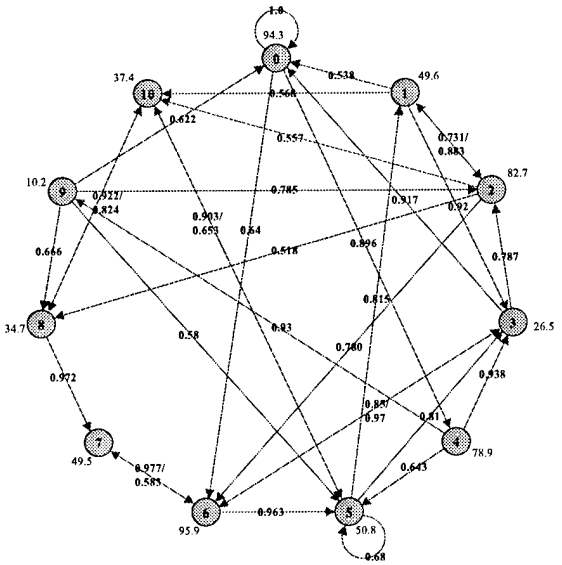


Figure 1 Graph Representation of the data collection problem.

The objective is to control the platforms in order to maximize the expected total value collected after N stages. Each platform begins at a base node (in this case, node 0 for all platforms) and may traverse one arc during each stage. There is a reward for each platform that has safely returned to its base node at the end of the Nth stage.

3. Limited Lookahead Policies

Consider a discrete-time dynamic system,

$$x_{k+1}=f_k(x_k,u_k,\omega_k),$$

where x_k is the state, u_k is the control to be selected from a finite set $U_k(x_k)$, and ω_k is a random disturbance. Denote the single-stage reward of control u from state x and disturbance ω by $g_k(x,u,\omega)$. A control policy $\pi=\{\mu_0,\mu_1,\ldots,\mu_{N-1}\}$ maps, for each stage k, a state x_k to a control value $\mu_k(x_k)\in U_k(x_k)$. There is a terminal reward $G(x_N)$ that depends on the terminal state x_N. The value-to-go of an optimal policy $\pi^*=\{\mu_0^*,\mu_1^*,\ldots,\mu_{N-1}^*\}$ starting from a state x_k at stage k can be computed using the following DP recursion

$$J_k^*(x_k)=\max_{u_k\in U_k(x_k)} E\{g_k(x_k,u_k,\omega_k)+J_{k+1}^*(f_k(x_k,u_k,\omega_k))\},$$

for all k and with the initial condition

$$J_N^*(x_N)=G(x_N).$$

For our problem, the state can be represented by a vector indicating for each node whether or not its value has been collected and by another vector indicating for each platform whether or not it is alive and if so, the node

at which the platform is located. The control at a particular stage provides for each platform that is alive a node that the platform is to attempt to visit during the current stage. If the platform successfully traverses the arc connecting its current node to the next node and the value of the node has not yet been collected, the current stage reward includes the value of the node. If the platform successfully reaches its base node during the last stage, there is a terminal reward associated with the platform.

Under a one-step lookahead policy, the control selected at stage k and state x_k is that which maximizes the following expression:

$$\max_{u_k \in U_k(x_k)} E\{g_k(x_k,u_k,\omega_k)+\tilde{J}_{k+1}(f_k(x_k,u_k,\omega_k))\},$$

where $\tilde{J}_{k+1}$ is some approximation of the value-to-go function J^*_{k+1}. Under a two-step lookahead policy, the control selected at stage k and state x_k is that which maximizes the above expression when $\tilde{J}_{k+1}$ is itself a one-step lookahead approximation; i.e., for all possible states $x_{k+1}=f_k(x_k,u_k,\omega_k)$, we have

$$\tilde{J}_{k+1}(x_{k+1})=\max_{u_{k+1}\in U_k(x_{k+1})} E\left\{\begin{matrix} g_{k+1}(x_k,u_k,\omega_k)+ \\ \tilde{J}_{k+2}(f_{k+1}(x_{k+1},u_{k+1},\omega_{k+1}))\end{matrix}\right\}.$$

Other multi-stage lookahead policies are similarly defined. Note that the number of lookahead stages, M, should be less than or equal to N-k-1. Essentially, the M-stage lookahead policy selects at stage k its decision by determining the optimal policy if there were only M stages remaining and the terminal cost was given by $E_{x_M}\{\tilde{J}_{k+M+1}(x_M)\}$, where x_M is the state resulting from applying the policy for the M decisions. A decision is selected, and the process is repeated at the next stage. The lookahead horizon is limited to the number of remaining stages, and so if the number of remaining stages is less than M, the M-stage lookahead policy determines the optimal strategy. A special case of such policies in which the value-to-go is approximated with zero is referred to in the literature as rolling or receding horizon procedures.

Generally, the effectiveness of limited lookahead policies depends on two factors:

1. The quality of the value-to-go approximation – performance of the policy typically improves with approximation quality.
2. The length of the lookahead horizon – performance of a policy typically improves as the horizon becomes longer (at least for small horizon lengths, e.g., 1-4).

However, as the size of the lookahead increases, the number of possible states that can be visited increases exponentially. To keep the overall computation practical, the complexity of the value-to-go approximation should be reduced for larger lookahead sizes. Balancing such tradeoffs is therefore a critical element in determining the size of the lookahead and the method for approximating the value-to-go. This paper explores several possibilities and tries to quantify the associated tradeoffs. One of the advantages of using limited lookahead policies for our particular problem is that the number of controls at a particular stage is fairly small and as a result, the computation required to explore all states that can be visited over the next M stages is manageable for small M.

3.1. Pruned Limited Lookahead Policies

Since the number of states that can be visited over M stages grows exponentially in M and also in the number of platforms, limited lookahead policies for M>1 are impractical for problems with many platforms. One approach to reducing the computation required for limited lookahead policies is to limit the number of states that can be visited. This can be accomplished by "pruning" controls that yield inferior intermediate values.

A pruned version of a limited lookahead policy depends on an integer parameter B that is typically selected through trial and error. In particular, we determine the one-step lookahead values for all controls available from our initial state. Controls that are not among those with one of the B best one-step lookahead values are pruned. We then repeat this process for each state that can be reached from a control that was not pruned and determine the one-step lookahead values for all controls available from these states. For each of these states, controls that are not among those with one of the B best one-step lookahead values are pruned. The number of times this process takes place is equal to the size of the lookahead.

Since the number of controls that are expanded from every state at every stage is limited, the computation required to find pruned policies is not exponential in the number of platforms. However, the computation is still exponential in the size of the lookahead.

4. Platform Decomposition

We now present an approach that involves exploiting the structure of our specific problem and decomposing it into a set of simpler problems. In particular, we decompose the problem into a separate sub-problem for each platform. This sub-problem consists of determining the optimal sequence of nodes, or path, to visit assuming that platform was the only one available. The optimal solution to each sub-problem can be found analytically. After a sub-problem is solved for a particular platform and before the next sub-problem is solved, the value of each

node in the associated path is updated to the value of the node multiplied by the probability that the node was not visited by the platform. This allows platforms to take into account paths assigned to previously scheduled platforms. When all of the sub-problems have been solved, a set of paths for each platform results. An outline of the platform decomposition approach is given below.

1. Assume that the platforms are ordered 1,2,...,V, and start with platform i=1.
2. Solve the single-platform problem optimally by finding a path or sequence of nodes $(n_{i1},n_{i2},...,n_{iN})$ that the platform should attempt to visit in order to maximize its expected value (which consists of collected node values plus the reward for the platform returning to the base station if n_N is the base node).
3. For every node in the path obtained in (2), scale the value of the node to 1 minus the probability that the node will be visited by platform i. This allows platforms that are scheduled later to take into account the path assigned to the current platform.
4. If i is less than the number of platforms, then let i=i+1 and go to (2). Otherwise, we are done.

The single-platform problem in step 2 can be solved using dynamic programming or by exhaustively considering all possible paths with N nodes. The computation required in either case is $O(D^N)$, where N is the number of stages and D is the average degree of a node. For sparsely connected graphs, the computation required is minimal.

The set of sub-problems can be solved once for a particular ordering of platforms or multiple times for various platform orderings. We will discuss several possibilities in the next section.

The platform decomposition heuristic yields for each platform i a path $(n_{ij},n_{i(j+1)},...,n_{iN})$, where j is the stage at which the heuristic is applied. This heuristic can be applied once before the mission begins to obtain a policy in which platform i attempts to visit node n_{ij} during the jth stage if it has not yet been destroyed. The heuristic can also be applied at every stage (for platforms that are still alive) using up-to-date state information, obtaining a policy in which platform i attempts to visit node n_{ij} during the jth stage. Finally, the heuristic can also be used to compute a value-to-go approximation for limited lookahead policies.

One of the main advantages to the platform decomposition approach is that the computation required is considerably smaller than limited lookahead policies. Assuming that the number of platform orderings considered remains fixed, the computation grows linearly in the number of platforms. In addition, as will be seen below, the method obtains solutions that are very close to the optimal. Unfortunately, while limited lookahead policies generalize easily to other problems, other problems may not have structures that easily decompose into sub-problems.

5. Computational Results

We now present some computational results from applying the above approaches to the problem described in Section 2. We consider a problem with N=10 stages, and either three or four platforms. The return rewards for the platforms were set to 12.7, 17.5, 19.2, and 55.0, and the most valuable platform was not included in the three-platform problems.

5.1. Limited Lookahead Policies

A limited lookahead policy consists of two main elements: the lookahead horizon, and the approximation of the value-to-go. We vary the size of the horizon from one to three and consider a number of approximations to the value-to-go. While there is some difference in the complexity of the value-to-go approximations, each one is straightforward to compute.

In many of our approaches, the value-to-go approximation for a particular state x after the first k stages, $\tilde{J}_k(x)$, involves heuristically generating for each platform i, a path or sequence of nodes $(n_{i(k+1)},n_{i(k+2)},...,n_{iN})$ to attempt to visit during the remaining N-k stages. We denote this collection of paths $P(x,k)$. Assuming each platform attempts to visit the nodes in its path, we can determine the expected collected value resulting from visiting nodes not visited during the first k stages:

$$C[P(x,k)]=\sum_{\substack{\text{nodes } n \text{ not}\\ \text{yet visited}}}\left(1-\prod_{\text{platforms } i}(1-p_{in})\right)c_n\ .$$

In the above equation, c_n is the one-time value associated with node n, and p_{in} is the probability that platform i visits node n:

$$p_{in}=\begin{cases}\prod_{j=k+1}^{l-1}p(n_{ij},n_{i(j+1)}), & \text{if } n_{il}=n \text{ for some } l,\\ 0, & \text{otherwise,}\end{cases}$$

where $p(n_{ij},n_{i(j+1)})$ is the probability of successfully traversing the arc connecting nodes n_{ij} and $n_{i(j+1)}$. To understand the expression for $C[P(x,k)]$, note that the term $\prod_{\text{platforms } i}(1-p_{in})$ provides the probability that none of the platforms successfully visits node n. The term

$\left(1-\prod_{\text{platforms } i}(1-p_{in})\right)c_n$ then provides the expected collected value at node n (the probability that at least one platform successfully visits the node multiplied by the node value).

We can also determine the expected reward resulting from platforms returning to the base node:

$$R[P(x,k)]=\sum_{\text{platforms } i} q_i v_i ,$$

where

$$q_i=\begin{cases}\prod_{j=k+1}^{N-1} p(n_{ij},n_{i(j+1)}), & \text{if } n_N \text{ is the base node,}\\ 0, & \text{otherwise,}\end{cases}$$

is the probability that platform i returns to the base node and v_i is the platform return reward.

The approximations to the value-to-go that we consider are given below. As can be seen in the descriptions, many of the approximations involve a combination of the expected collected node value, $C[P(x,k)]$, and the expected platform return reward, $R[P(x,k)]$, assuming each platform attempts to visit the nodes in the paths specified in $P(x,k)$.

1. The first approach approximates the value-to-go with zero:

   $$\tilde{J}_k(x)=0.$$

2. The second approach approximates the value-to-go with the sum of the expected collected node value and the expected platform return reward collected over a set of greedy paths:

   $$\tilde{J}_k(x)=C[P_g(x,k)]+R[P_g(x,k)].$$

   The nodes along the greedy path for platform i, $(n_{i(k+1)},\ldots,n_{iN})$, are determined as follows:

   $$n_{i(j+1)}=\arg\max_{n\in\eta(n_{ij})}\{p(n_{ij},n)\cdot c_n\},$$

   where $\eta(n_{ij})$ is the set of nodes that can be reached from node n_{ij}, and n_{ik} is the node at which platform i is located after k stages.

3. The third approach approximates the value-to-go with the expected platform return reward collected over the set of "safest" paths:

   $$\tilde{J}_k(x)=R[P_s(x,k)].$$

   The safest path is that which yields the highest probability of a platform returning successfully to its base node. These paths can be computed apriori using dynamic programming. (Essentially, the computation is equivalent to solving a set of shortest path problems.)

4. The fourth approach approximates the value-to-go with the sum of the expected collected node value and the expected platform return reward collected over the set of safest paths:

   $$\tilde{J}_k(x)=C[P_s(x,k)]+R[P_s(x,k)].$$

5. The fifth approach approximates the value-to-go with the sum of the expected collected node value and the expected platform return reward collected over the set of "most valuable" paths:

   $$\tilde{J}_k(x)=C[P_m(x,k)]+R[P_m(x,k)].$$

   The most valuable path is that which yields the highest expected total value that could be attained by a single vehicle during the remaining stages assuming none of the values at any of the nodes have yet been collected. These paths can also be computed apriori using dynamic programming.

6. The sixth approach combines (4) and (5). The value-to-go is approximated with the maximum of the values determined by those approaches.

Table 1 provides the expected optimal values for the problem illustrated in Figure 1 for a three-platform problem and a four-platform problem. We have computed these values using dynamic programming, and the computation required for the four-platform problem was approximately one week on a Sun Ultra 60 workstation. Table 1 also provides the results of applying a greedy algorithm, in which each platform selects as its next node that which maximizes its expected collected value for that stage, to one thousand sample trajectories. The performance achieved in our earlier efforts of applying rollout strategies using 20 or more Monte Carlo simulations ranged on average from 600 to 610 for the four-platform problem.

Table 1 The expected optimal values and the results of applying the greedy algorithm for the three and four platform problems.

# Platforms	Expected Optimal	Greedy
Three	574.5	475.72
Four	641.0	533.89

Tables 2 and 3 provide the values averaged over one thousand sample trajectories by applying the limited lookahead polices for lookahead sizes of one to three, using the six value-to-go approximations described above. The particular approximation approach used is given in the leftmost column. As can be seen, while the 2-stage policies generally provided results that improved significantly upon those of the 1-stage policies, those of the 3-stage policies were not substantially better and in a few cases were worse than those of the 2-stage policies.

The sixth value-to-go approximation seemed to yield slightly better results than the other approximations. However, the third through sixth approximations were basically comparable. Overall, these approaches improved significantly upon the greedy algorithm and were able to obtain values close to the optimal for lookahead sizes greater than one. For lookahead sizes greater than one, these approaches were also able to obtain results slightly better than those obtained using rollout strategies with Monte Carlo simulations.

Table 2 The results of applying the limited lookahead policy to the three-platform problem.

Value-to-go Approximation	1-stage	2-stage	3-stage
1	491.09	539.58	543.40
2	520.57	543.95	553.74
3	506.55	550.82	553.76
4	500.69	529.98	559.46
5	554.10	557.94	561.75
6	555.97	563.45	561.09

Table 3 The results of applying the limited lookahaead policy to the four-platform problem.

Value-to-go Approximation	1-stage	2-stage	3-stage
1	543.32	574.24	582.75
2	589.56	607.31	593.08
3	581.48	613.29	615.63
4	574.06	618.55	619.45
5	582.84	594.09	606.96
6	595.44	615.10	624.16

Tables 4 and 5 provide the average values obtained over the same thousand sample trajectories by applying the pruned limited lookahead polices for lookahead sizes of two and three, using the value-to-go approximations described above. (Note that a pruned one-step lookahead policy is equivalent to the fully expanded one-step lookahead policy.) As can be seen, the results of these approaches do not vary significantly from the fully expanded lookahead policies. In some cases, the pruned policies performed one or two percent worse and in other cases, they performed one or two percent better.

Table 4 The results of applying the pruned limited lookahead policy to the three-platform problem.

Value-to-go Approximation	2-stage	3-stage
1	538.56	523.48
2	532.70	551.10
3	550.82	553.56
4	552.47	559.46
5	556.38	555.47
6	561.22	563.82

Table 5 The results of applying the pruned limited lookahead policy to the four-platform problem.

Value-to-go Approximation	2-stage	3-stage
1	573.19	575.21
2	605.57	607.21
3	608.98	616.21
4	613.23	615.38
5	595.55	592.50
6	613.49	617.04

5.2. Platform Decomposition Results

In applying platform decomposition to our problem, we considered the following approaches to ordering the platforms:

1. A single ordering in ascending order of the platform return reward.
2. All possible orderings.
3. A "rollout" of the ordering in (1) as described by Bertsekas, Tsitsiklis and Wu ([4]). I.e., assuming that the first *i*-1 platforms have been selected, the *i*th platform is determined as follows:
   i. Consider each remaining platform in turn as the next platform and leave the other vehicles in their original order.
   ii. Solve the set of single-platform problems in the given order.
   iii. Select as the *i*th platform that which yields the best result.

As mentioned in Section 4, there are several ways to apply the heuristic:

- The heuristic can be applied once to obtain a policy for all stages.
- The heuristic can be applied at every stage to obtain a control for the current stage using current state information.
- The heuristic can be used to generate a value-to-go approximation for a limited lookahead policy.

Table 6 provides the average values obtained over the same thousand sample trajectories by the platform decomposition approach. The result of applying the heuristic for all possible orderings and following the paths obtained for all stages is provided in the first row. The next three rows provide the results when the heuristic using the three orderings described above (least expensive to most expensive, all possible orderings, and a rollout of the orderings) is reapplied at every stage to obtain the current control. The remaining rows provide the results when the heuristic is used to provide a value-to-go approximation for a one-stage limited lookahead policy using the orderings described above is used. As can be seen, these approaches performed extremely well. The heuristic alone performed comparably to 2-stage lookahead policies, and the other variations were able to obtain strategies that yielded results that were less than one percent from the optimal expected results.

Table 6 The results of applying platform decomposition approaches. The first row provides the result of applying the heuristic for all possible platform orderings before the start of the mission and following the resulting paths. The next three rows provide the results of reapplying the heuristic at every stage using various platform orderings (1: least expensive to most expensive; 2: all possible orderings; 3: a rollout of the orderings). The last three rows provide the results of applying one-stage limited lookahead policies using the values obtained from the platform decomposition heuristic (under the various platform orderings) as an approximation to the value-to-go.

	3 platforms	4 platforms
Heuristic alone	550.85	608.89
Heuristic reapplied-1	568.81	634.97
Heuristic reapplied-2	573.83	637.81
Heuristic reapplied-3	573.83	637.81
1-stage LL-1	570.97	633.04
1-stage LL-2	571.29	635.65
1-stage LL-3	571.29	635.65

5.3 Computation Times

The following table provides the average on-line computation time (in seconds) to apply the approaches described above to one hundred sample trajectories of the four-platform problem. The off-line computation time for the limited lookahead policies was negligible. We have measured the time required to compute the controls. In practice, this time is critical since it must be within the real-time constraints of the problem. The table gives the total time to compute these controls for the ten stages. Since these times depend on the state trajectory of the system, which is random, we averaged over 100 trajectories and recorded the results in Table 7. The times for the one-stage lookahead have not been included as the time required was negligible. The experimental results were conducted on a Sun Ultra 60 workstation. As can be seen from the table, the pruned lookahead policies were significantly faster than the fully expanded lookahead policies. Considering this in combination with the fact that the performances of the two versions are comparable suggests that that pruned lookahead policies may be more useful in practice. The pruned lookahead policies were also generally much faster than the rollout algorithms using Monte Carlo simulations, whose computation times varied from 5 to over 300 seconds per sample trajectory. The decomposition approaches were extremely fast, and also provided the best results. Reapplying the decomposition heuristic at every time step appears to be the best option. However, it is not clear how easily such approaches can be applied to variations of the problem.

Table 7 Time to compute the controls for ten stages under the various approaches averaged over 100 sample trajectories of the four-platform problem. The first six lines provide the times corresponding to the fully expanded and pruned limited lookahead results given in Tables 3 and 5. The next six lines provide the times corresponding to the last six platform decomposition results given in Table 6.

	2-stage lookahead		3-stage lookahead	
	Full	Pruned	Full	Pruned
LL-1	0.77	0.12	120.4	1.8
LL-2	9.41	1.04	1358	16.4
LL-3	1.35	0.22	134.7	3.4
LL-4	6.16	0.71	716.5	9.4
LL-5	9.16	0.91	796.5	8.2
LL-6	14.85	1.73	1258	14.5
PD-Heuristic reapplied-1			0.41	
PD-Heuristic reapplied-2			9.42	
PD-Heuristic reapplied-3			1.62	
PD-1-stage LL-1			51.50	
PD-1-stage LL-2			396.52	
PD-1-stage LL-3			138.04	

6. Summary

In this paper, we have considered alternatives to using on-line simulations for approximating the value-to-go for adaptive multi-platform scheduling in a risky environment. The main limitation to using rollout algorithms with on-line simulations that was determined in

our previous effort was the amount of computation required to evaluate control options at every stage. We instead considered two alternatives.

The first approach involved examining control options over a limited horizon. In our experimental results, this method produced results that were slightly better than those obtained through rollout algorithms with on-line simulations with similar computation time. Computation time was reduced significantly by introducing a pruning technique without loss in performance.

The second approach involved decomposing the problem into sub-problems associated with each platform. This method produced results that were extremely close to the optimal values and required small computation times. However, while limited lookahead methods generalize well to other problems, the decomposition method requires a suitable problem structure. Furthermore, this method may not perform well for problems with an appropriate structure if the decomposed elements require significant coordination.

7. References

[1] Alden, J.M., Smith, R.L., "Rolling Horizon Procedures in Nonhomogeneous Markov Decision Processes," *Operations Research*, V. 40, 1992.

[2] Bertsekas, D.P., Castañon, D.A., "Rollout Algorithms for Stochastic Scheduling Problems," *Journal of Heuristics*, V. 5, 1999.

[3] Bertsekas, D.P., Castañon, D.A., Curry, M.L., Logan, D., "Adaptive Multi-platform Scheduling in a Risky Environment," *1999 Proceedings from Advances in Enterprise Control Symposium*, Nov 1999.

[4] Bersekas, D.P., Tsitsiklis, J.N., Wu, C., "Rollout Algorithms for Combinatorial Optimization," *Journal of Heuristics*, V. 3, 1997.

Adaptive Multi-platform Scheduling in a Risky Environment*

Dimitri P. Bertsekas
Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA 02139

Michael L. Curry
ALPHATECH, Inc. 50 Mall Road Burlington, MA 01803

David A. Castañon
Dept. of Electrical and Computer Engineering,Boston University, Boston, MA 02215

David Logan
ALPHATECH, Inc. 50 Mall Road Burlington, MA 01803

Abstract

In this paper, we investigate the use of rollout algorithms for adaptive multi-platform scheduling in a risky environment. The underlying decision problem is motivated by several Air Force applications: data collection, sensor management, and air operations planning. These problems may be solved optimally with stochastic dynamic programming (SDP), but have overwhelming computational requirements. Rollout algorithms reduce computational requirements by using on-line learning and simulation to approximate SDP with a base heuristic. While they do not aspire to optimal performance, rollout algorithms typically result in a consistent and substantial improvement over the underlying heuristics. A multi-platform planning and scheduling problem is used to demonstrate rollout performance.

1 Introduction

The planning and execution of multiple missions in the presence of risk is a problem which arises in many important military contexts. In data collection applications, multiple UAV platforms may be tasked to interrogate different areas, with the risk of platform destruction as each platform pursues its collection mission. In attack air operations, multiple platforms follow risky trajectories to attack enemy targets. For both applications, sensors and communication equipment can provide up-to-date information concerning individual mission and platform status, and thus provide notification of platform losses. This creates opportunities for retasking surviving platforms in order to best achieve mission objectives.

In mathematical terms, the above class of problems can be viewed as a sequential decision problem, where each decision is based on the observation of certain discrete events. These decisions affect the evolution of a system state (mission), which is also influenced by random discrete events (e.g. platform destruction). The goal is to select the current decisions as a function of the current system state, in a manner that optimizes mission performance.

The above class of problems can be formulated as Markov decision problems [3],[5]. The principal approach for solving such problems is dynamic programming (DP), which selects feedback rules to determine optimal controls for each possible state. These optimal controls are determined by evaluating at each stage the immediate expected cost of the current decision, plus the future optimal cost-to-go over future decisions. However, it is well known that computation of the optimal cost-to-go for each future state is computationally intractable for all but the simplest of problems, making direct application of DP an impossible task for multi-platform control.

In recent years, there has been a great deal of research on approximate DP methods based on computing suitable approximations to the optimal cost-to-go. These methods are collectively known as neuro-dynamic programming (NDP) [1]. In NDP, the optimal cost-to-go is approximated by a parametric function; critical issues for NDP include the selection of the parametric class of approximating functions, and selection of the approximating parameters.

* This work was supported by the Air Force Office of Scientific Research under contract #F49620-98-C-0023.

In this paper, we apply a particular class of NDP algorithms, known as rollout algorithms [2], to risky multi-platform planning and scheduling problems. Rollout algorithms are a form of NDP which exploits knowledge of suboptimal heuristic decision rules to obtain approximations to the optimal cost-to-go for use in NDP. We develop different rollout algorithms for risky multi-platform scheduling, and illustrate the relative performance of the rollout algorithms and the original suboptimal decision rules in the context of a specific example. The results illustrate that significant performance improvements can be obtained using rollout algorithms, with a modest increase in computation complexity.

2 Illustrative Overview

To illustrate the types of problems of interest and results developed in this paper, consider the data collection problem illustrated in Figure 1. There are several data collection assets, which may travel to examine targets. There is a value associated with collecting the information on each target. Platforms also run the risk of destruction while performing collection on a asset, due to the presence of local defenses.

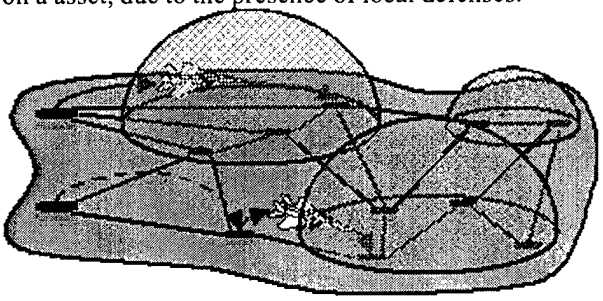


Figure 1 Illustration of Data Collection Problem

Ideally, each data collection asset will be provided a schedule of targets for information collection, which is coordinated among assets to ensure maximal value collected. However, due to the risk inherent in the collection process, platforms can be destroyed, and thus the original schedules should be adapted whenever a destruction event occurs in order to recover the most collection value. If these abrupt events are not anticipated in the original schedules, the possible modifications to the schedules may be so constrained that highly sub-optimal performance results.

The basic theory of dynamic programming provides a framework for developing schedules which anticipate the future occurrence of contingencies such as platform destruction, and hedge the selected schedules in anticipation of needed retasking. Thus, the resulting schedules can be adapted to contingencies with minimal performance degradation, resulting in robust, stable control.

The computational requirements of DP depend on the number of future states required to describe the system. To illustrate the number of states required, assume that there are N targets, M collection assets, and that we simplify physical position descriptions to describe only the N positions of the targets. Then, the number of possible combinations of positions is M^N, and the number of possible uncollected target sets at a given time is 2^N, resulting in numbers of states $(2M)^N$. For modest numbers of assets and targets, the number of states far exceeds our capability for computing and/or storing the resulting optimal decision rules.

Using NDP principles such as rollout strategies greatly reduces the resulting computational complexity. DP considers all of the possible states and computes a tentative decision for each possible state, whereas NDP only computes decisions for states that actually occur in the scenario. Thus, the number of states considered by NDP considered is much smaller, but can only be determined in real-time. In the rollout methodology, once the scenario reaches a given state where a contingency has been observed, new plan options are evaluated in real-time to select the future actions. The result is a practical algorithm for feedback control in complex multi-platform planning and scheduling applications. The fundamental questions about this approach are how good is the performance achieved, and how much real time computation is required. These questions are explored in greater detail in the subsequent sections.

3 Rollout Algorithms

Consider a discrete-time version of a dynamic decision problem,

$$x_{k+1} = f(x_k, u_k, \omega_k)$$

where x_k is the state, u_k is the control to be selected from a finite set $U(x_k)$, and ω_k is a random disturbance. Denote the single-stage cost of control u from state x and disturbance ω by $g(x,u,\omega)$.

A control policy $\pi = \{\mu_0, \mu_1, \cdots\}$ maps, for each stage k, a state x to a control value $\mu_k(x) \in U(x)$. In the N-stage horizon problems considered herein, k takes values $0,1,\cdots,N-1$, and there is also terminal cost $G(x_N)$ that depends on the terminal state x_N. The cost-

to-go of policy π starting from a state x_k at time k can be computed using the following DP recursion

$$J_k^\pi(x) = E\{g(x,\mu_k(x),\omega) + J_{k+1}^\pi(f(x,\mu_k(x),\omega))\} \quad (1)$$

for all k and with the initial condition

$$J^\pi{}_N(x) = G(x)$$

The rollout policy based on π is denoted by $\overline{\pi} = \{\overline{\mu}_0, \overline{\mu}_1, \cdots\}$, and is defined by the operation

$$\overline{\mu}_k(x) = \arg\min_{u \in U(x)} E\{g(x,u,\omega) + J_{k+1}^\pi(f(x,u,\omega))\} \quad (2)$$

for all x and k. Thus the rollout policy selects decisions by balancing the current cost with future costs-to-go, where the optimal costs-to-go are approximated by the performance of the base policy π.

A straightforward approach for computing the rollout control at a given state x and time k is to use Monte Carlo simulations of the base policy. To implement this approach, we consider all possible controls $u \in U(x)$ and generate a "large" number of simulation trajectories of the system starting from x, using u as the first control, and using the policy π thereafter. Thus the simulated trajectory has the form

$$x_{i+1} = f(x_i, \mu_i(x_i), \omega_i) \quad i = k+1, \cdots N-1$$

where the first generated state is

$$x_{k+1} = f(x, u, \omega_k)$$

The costs corresponding to these trajectories are averaged to obtain the Q-factor

$$Q(x,u) = E\{g(x,u,\omega) + J^\pi{}_{k+1}(f(x,u,\omega))\}$$

In reality, only an approximation $\tilde{Q}(x,u)$ is obtained because of the associated simulation error. The approximation becomes increasingly accurate as the number of simulation trajectories increases. Once the approximate Q-factor $\tilde{Q}(x,u)$ corresponding to each control $u \in U(x)$ is computed, we obtain the approximate rollout control $\tilde{\mu}_k(x)$ by the minimization

$$\tilde{\mu}_k(x) = \arg\min_{u \in U(x)} \tilde{Q}_k(x,u)$$

4 Example: Data Collection Problem

The graph in Figure 2 corresponds to an example data collection problem. Each node represents a geographical area of interest with a one-time value (i.e., data may only be collected once from each location). The arcs represent connectivity among the geographical regions and may be successfully traversed with a known probability. Platforms traverse the graph and collect data (value) at each node, or else they are destroyed while traversing specific arcs. If a platform is destroyed on an arc, the value of the destination node is not collected, which can result in retasking other platforms.

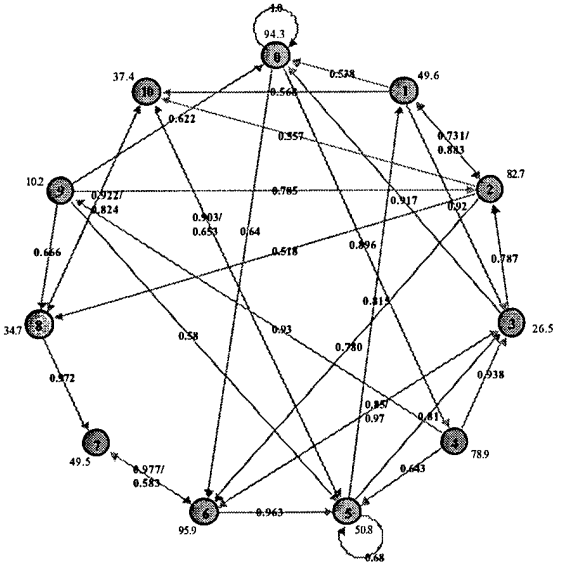


Figure 2 Graph Representation of the Data Collection Problem

The objective is to control the platforms in order to maximize the expected total value collected after N stages ($N = 10$ will be used). Each platform begins at a base node (in this case, node 0 for all platforms) and may traverse one arc during each stage. If a platform does not return to its base node within N stages, there is a penalty associated corresponding to platform loss.

4.1 The Base Policy: Greedy

As a base policy for rollout, we use the greedy policy $\pi = \{\mu_0, \mu_1, \cdots\}$, which is defined by the operation

$$\mu_k(x) = \arg\max_{u \in U(x)} E\{g(x,u,\omega)\}$$

for all x and k. The control u is a vector of locations corresponding to the next destination of each platform. Similarly, each element of $\mu_k(x) = [\mu_k^0(x), \mu_k^1(x), \cdots]$ corresponds to a specific platform.

To reduce the computational overhead, we consider the platforms sequentially. The control for the first platform, $\mu_k^0(x)$, is selected independent of the other platforms' controls as:

$$\mu_k^0(x) = \max_{u \in U^0(x)} E\{g(x,u,\omega)\}$$

where $U^0(x)$ are feasible controls for platform 0. The control, $\mu_k^j(x)$, for subsequent platforms is conditioned

on all the previously selected controls $\mu_k^0(x), \mu_k^1(x), \cdots, \mu_k^{j-1}(x)$ and defined by the operation

$$\mu_k^j(x) = \max_{u \in U^j(x)} E\left\{g(x,u,\omega) \middle| \mu_k^0(x), \cdots, \mu_k^{j-1}(x)\right\}$$

This allows the greedy policy to anticipate the arrival of platforms at specific nodes based on previously selected controls.

The greedy policy also forces platforms to return within N stages by constraining the set $U_k(x)$ of feasible controls to those for which a return within N stages is possible

The performance of the greedy policy corresponds to the cost-to-go from the initial state x_0.

$$J^{\pi}{}_0(x_0) = E\left\{G(x_N) + \sum_{i=0}^{N-1} g(x_i, \mu_i(x_i), \omega_i)\right\}$$

where the expectation is taken over simulation trajectories of the form

$$x_{i+1} = f(x_i, \mu_i(x_i), \omega_i) \quad i = 0, \cdots, N-1$$

The performance of the greedy policy provides a baseline for evaluating the rollout policy.

4.2 Rollout Algorithm

The rollout policy is computed using the greedy policy as its base policy, as indicated in equations (1-2). The performance of the rollout policy is evaluated in a manner similar to the greedy policy, by using the cost-to-go from the initial state x_0.

$$J^{\overline{\pi}}{}_0(x_0) = E\left\{G(x_N) + \sum_{i=0}^{N-1} g(x_i, \overline{\mu}_i(x_i), \omega_i)\right\}$$

with the simulation trajectories

$$x_{i+1} = f(x_i, \overline{\mu}_i(x_i), \omega_i) \quad i = 0, \cdots N-1$$

To reduce the relative variance of performance values, we use the same simulation trajectories in the evaluations of all policies.

One drawback of this approach is that many on-line Monte Carlo simulations may be required to compute the rollout decision at a state. As an alternative, we can use approximations trained with off-line simulations, as discussed in the next subsection.

4.3 Rollouts and Neural Approximations

To reduce the on-line computational overhead of the rollout policies, we propose to train off-line a parametric approximation of the greedy policy performance based on features which characterize the current state. In particular, the features that we use correspond to the values achieved by the greedy policy under a small number of certainty-equivalence scenarios, which capture the graphical dependence of the scheduling problem. This approach was initially proposed in [2].

To compute a feature at a given state x_k at time k, we fix the remaining disturbances at some nominal values $\overline{\omega}_k, \overline{\omega}_{k+1}, \cdots, \overline{\omega}_{N-1}$, and generate a state and control trajectory of the system using the base policy π starting from x_k and time k. The corresponding cost is denoted by $\tilde{J}_k^{\pi}(x_k)$, and is a feature which is used to estimate the true cost $J_k^{\pi}(x_k)$. We use a small number of disturbance trajectories corresponding to different scenarios. The feature values computed for each of these scenarios are combined parametrically to approximate the cost of the base policy using the functional form:

$$\tilde{J}_k(x_k, r) = r_0 + \sum_{m=1}^{M} r_m C_m(x_k) \qquad (3)$$

where $r = (r_0, r_1, \cdots, r_M)$ is a vector of parameters to be determined, and $C_m(x_k)$ is the cost corresponding to the m^{th} scenario. The parameters r are determined by an off-line training process using simulations of the base policy. Equation (3) can then be used on-line, computing the costs $C_m(x_k)$, to evaluate the base policy cost from state x_k at time k.

5 Experimental Results

A series of experiments were performed on the example problem presented in section 4.1, evaluating the performance of the base greedy policy and different variations of rollout algorithms.

The greedy heuristic used for the baseline policy is based on an objective function with two terms, one associated with the achievable value of data collected and the other associated with the potential loss of the vehicle. These values depend on probability ratios associated with risk. The objective function for vehicle k at state x_i is given by

$$g_k(u_i, x_i, \omega) = \frac{p_{ij}}{(1-p_{ij})} n_j(x_i) - \frac{(1-p_{ij})}{p_{ij}} v_k$$

where $n_j(x)$ is the achievable value of option j given the current state, v_k is the value of vehicle k, and p_{ij} is the transition probability associated with option j (p_{ij} characterizes the disturbance ω). This objective function is a risk neutral strategy that computes the marginal difference between the largest acceptable loss

and the smallest acceptable gain associated with option j.

The greedy heuristic is evaluated by determining the cost-to-go from the initial state x_0.

$$J^{\pi_0}(x_0) = E\left\{G(x_{10}) + \sum_{i=0}^{9} g(x_i, \mu_i(x_i), \omega_i)\right\}$$

where the expectation is approximated with 100 Monte Carlo simulation trajectories of the form

$$x_{i+1} = f(x_i, \mu_i(x_i), \omega_i) \quad i = 0, \cdots, 9$$

The evaluation of the greedy heuristic resulted in an estimated value of 569.3 (the standard deviation of the estimate is 12.2). As a benchmark, the total value achievable is 714, arising from:

Total Collectible Value	610
Total Vehicle Value	104

We conducted three types of experiments, exploring different rollout options. The first set of experiments used different number of Monte Carlo runs in the rollout algorithm to evaluate the relative performance of the different controls for each state considered. Tested conditions ranged from 5 to 40 Monte Carlo experiments per decision.

The second set of experiments evaluated alternatives in the planning horizon considered in the rollout problem. It has been conjectured [2] that the performance of rollout strategies degrades after increasing the planning horizon beyond a threshold, due to the approximation of the optimal future policy by a base policy. This approximation becomes less accurate with increasing planning horizon. To test this, we conducted experiments where we varied the horizon used by the rollout policy to evaluate the base policy.

The final set of experiments compares the performance of the Monte Carlo rollout algorithms with the performance of the algorithms based on parametric function approximations using certainty equivalence features.

5.1 Variations in Monte Carlo Runs

In these experiments, the number of Monte Carlo runs used to evaluate the performance of the base policy in the rollout algorithm varies from 5, 10, 20, 30, and 40 Monte Carlo runs. For each of these experiments, evaluation of the rollout policy performance is conducted in a manner identically to the evaluation of the greedy policy performance described previously. That is, 100 independent Monte Carlo simulation trajectories are used, of the form

$$x_{i+1} = f(x_i, \bar{\mu}_i(x_i), \omega_i) \quad i = 0, \cdots 9$$

where $\bar{\mu}_i(x_i)$ is the rollout control policy defined by the operation

$$\bar{\mu}_i(x) = \arg\max_{u \in U(x)} E\{g(x,u,\omega) + J^{\pi}{}_{i+1}(f(x,u,\omega))\}$$

and this last expectation is also approximated with either 5, 10, 20, 30, and 40 Monte Carlo simulations.

The results of this experiment are shown in Figure 3. As the results indicate, the performance achieved by the rollout strategies using 20 or more Monte Carlo simulations range on average from 600 to 610, a range which is far superior to the greedy policy performance average of 569.3. The results suggest that a modest number of simulations are required to select good controls in this example.

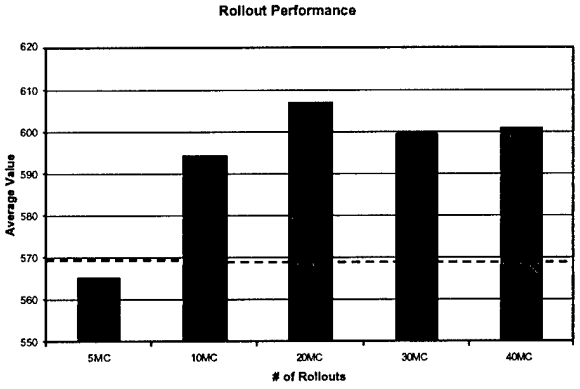


Figure 3 Rollout Performance

5.2 Variations in Planning Horizon

The rollout algorithm performs a single policy improvement step on the greedy heuristic. This allows the rollout trajectory to deviate significantly from the greedy trajectory, especially over long horizons. In this case, the cost-to-go estimate derived from greedy trajectories may not reflect the actual cost-to-go of the rollout algorithm. One way of avoiding this problem is to evaluate the future cost-to-go of the base policy over a limited horizon. Figure 4 shows the performance of this rollout algorithm with various horizons, where 20 Monte Carlo experiments are used to evaluate each policy.

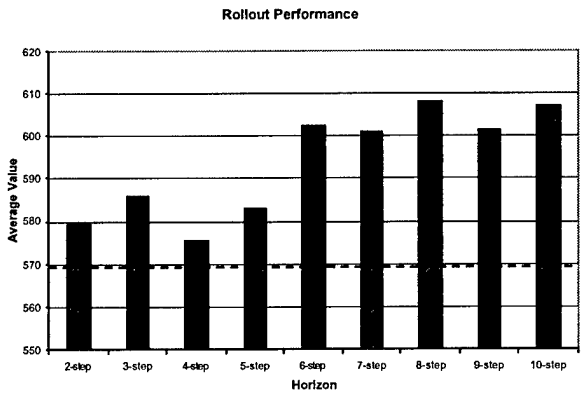


Figure 4 Performance as a Function of Horizon

The results in Figure 4 do not support the conclusion that there is a maximum planning horizon beyond which the rollout performance degrades. However, the results need closer examination to understand whether the use of a different base heuristic would exhibit similar behavior.

5.3 Off-Line Training vs Monte Carlo

In these experiments, we compare the performance of rollout algorithms based on the parametric approximations of Section 4.3 with the Monte Carlo rollout algorithms of Section 4.2. The parametric approximations were based on certainty equivalent features, which corresponded to selecting specific threshold values and declaring all arcs with probabilities of survival greater than the threshold to be safe, and all arcs with probabilities of survival less than or equal to the threshold to have a certainty of destroying any vehicles on those arcs. The resulting graph is a deterministic graph, which leads to fast evaluation of the base policy. The performance obtained for different values of thresholds provided the base features for the parametric approximation.

Several rollout algorithms were evaluated: First, we used algorithms based on single features, with trivial parametric approximation. Second, we used algorithms using weighted combinations of features, with weights trained off-line using training data. Finally, we used optimized weights, searching in the space of possible weights for optimal performance; this is not a practical algorithm, but provides a baseline for the achievable performance from training algorithms.

The experimental results are summarized in Figure 5. These experiments used only two features in the parametric interpolation, corresponding to two different values of thresholds. In Figure 5, four pairs of features are considered along the x-axis. For each pair of features, the rollout performance is evaluated with optimal weights, and with trained weight. Rollout performance is also evaluated for each of the features used in isolation.

In Figure 5, the upper dashed line represents the best performance achieved using the Monte Carlo rollout approach, and the lower dashed line represents the performance of the greedy heuristic. The dotted line indicates the performance using two statistically determined features combined with equal weights.

The results of Figure 5 are surprising, in that the algorithms based on off-line training seldom approach the performance achieved by the optimal weighted combination. Figure 5 shows that the rollout algorithm based on features with trained weights was not able to offer a consistent and significant improvement over the greedy heuristic. In some cases the combination of features with trained weights were not able to perform as well as the features individually.

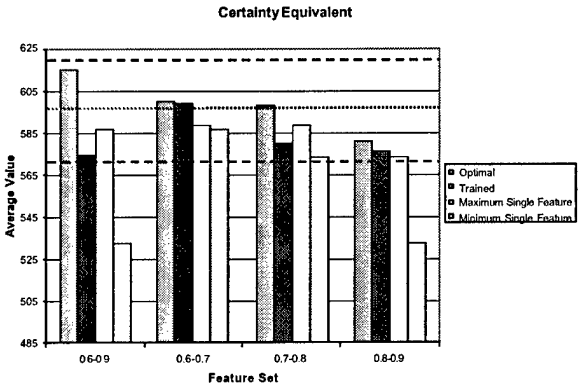


Figure 5 Rollout Performance with Feature Pairs

Figure 5 shows that the optimally selected weights using the features 0.6 and 0.9 achieved an overall performance close to that of the Monte Carlo approach. However, when other pairs of features were used, the performance was significantly worse. An alternative to training or optimization is to select the parameters analytically. The dotted line in Figure 5 shows the performance of a combination of features that were selected with equal weights to "match" the statistical distribution (mean and variance) of risk within the problem. This approach provides a significant improvement over the greedy heuristic without the computational cost of training or optimization. This approach appears worthy of further investigation due to its simplicity.

In sum, rollout algorithms using parametric approximations did not perform as well as rollout algorithms using Monte Carlo simulations.

6 Conclusions

In this paper we have considered the use of rollout algorithms for adaptive multi-platform scheduling in a risky environment. We explored different variations of rollout algorithms, using combinations of on-line Monte Carlo simulation and parametric approximations. Our experimental results show that rollout algorithms using on-line simulation perform significantly better than the reference base heuristic policies, using only a modest number of Monte Carlo trajectories.

In our experiments, we found that rollout algorithms based on parametric approximations to the cost-to-go failed to achieve the level of performance of similar rollout algorithms using on-line Monte Carlo simulations. The parametric approximations suffered from two limitations: First, the training techniques often failed to identify the best weight combinations. Second, the parametric approximations were unable to generalize accurately across the broad class of states which occurred in the problem. Our experiments were limited to simple classes of parametric approximations using the concept of certainty equivalence scenarios. Exploration of alternative approximations using different features is an area for future investigations.

The main limitation of the Monte Carlo rollout algorithms is the amount of on-line computation required to evaluate the different options at each state. We are currently investigating techniques based on discrete-event systems and perturbation analysis [4] to reduce the number of simulations required to evaluate multiple alternatives.

7 References

[1] Bertsekas, D.P., Tsitsikis, J.N., *Neuro-Dynamic Programming*, Athena Scientific, 1996.

[2] Bertsekas, D.P., Castañon, D.A., "Rollout Algorithms for Stochastic Scheduling Problems," *Journal of Heuristics*, V. 5, 1999.

[3] Bertsekas, D.P., *Dynamic Programming and Optimal Control*, Athena Scientific, 1995

[4] Ho, Y.C., Cassandras, C.G., Chen, C.H., Dai, L., "Ordinal Optimization and Simulation," submitted to special issue on simulation to be published by INFORMS 1999.

[5] Ross, S. M., *Introduction to Stochastic Dynamic Programming*, Academic Press, N.Y., 1983.

Closed-Loop Control for Joint Air Operations

Jerry M. Wohletz, David A. Castañon, and Michael L. Curry

Abstract— **This paper focuses on the problem of providing real-time, closed-loop feedback control of Joint Air Operations (JAO) via near-optimal mission assignments. For this application, a rollout algorithm is employed which is based on the theory of stochastic dynamic programming. The primary benefits of this technology are agile and stable control of distributed stochastic systems. The rollout algorithm is applied to a small JAO scenario that includes limited assets, risk/reward that is dependent on mission composition, basic threat avoidance routing, and multiple targets, some of which are fleeting and emerging. Simulation results illustrate the benefits of the closed-loop feedback control. It is shown that the rollout strategy provides statistically significant performance improvements over an open-loop feedback strategy that uses the same baseline heuristic. The performance improvements are attributed to the fact that the rollout algorithm was able to learn near-optimal behaviors that were not modeled in the baseline heuristic.**

Keywords— **Large-scale control, approximate dynamic programming, stochastic systems, adaptive control.**

I. Introduction

CURRENTLY , air operations are executed according to an Air Tasking Order (ATO) which is developed every 24 hours. If you included the end-to-end development time — the air operations planning, tasking, and executing — the process takes 72 hours. Given the dynamic nature of air operations where things can change in a matter of minutes, this open-loop control strategy suffers from a lack of agility. Moreover, given our current information gathering capability, our awareness of the battle space has never been greater. The goal of this research is to take advantage of the battle space information and develop closed-loop control strategies to improve the agility of air operations.

Ideally, air packages are assembled and assigned to targets such that their coordinated effect efficiently achieves a campaign objective with the available resources. However, due to the inherently uncertain JAO environment, i.e. popup threats, time critical targets, asset destruction, etc., the original tasking should be adapted whenever a significant event occurs in order to achieve the campaign objective. If these abrupt events are not anticipated, the possible modifications may be so constrained that significant performance degradation results.

In this paper, the JAO problem is view as a stochastic control problem and an Approximated Stochastic Dynamic Programming (ASDP) technique known as the Rollout Algorithm (RA) is applied. This control strategy anticipates future significant events, and hedges assets for the opportunity of recourse. Thus, the resulting missions can be adapted to contingencies with minimal performance degradation, resulting in robust, stable control.

In Section II, an overview of the control methodology is presented. Simulation results for a small JAO scenario are presented in Section III. Finally, conclusions are presented in Section IV.

II. Methodology

The JAO environment is an uncertain dynamical system that has the following attributes: control decisions made over time; probabilistic transition from one state to the next, which is dependent on the choice of control; and risk/rewards that are accumulated during each transition, which is dependent on control and state transition outcome. Thus, the tasking of air packages in a JAO environment can be viewed as a sequential decision problem where each decision is based on the observations of certain discrete events.

This class of problems can be formulated as a Markov decision problem [1]. The principal approach for solving such problems is Stochastic Dynamic Programming (SDP). Using the SDP formulation, an optimal control solution is computed off-line, and on-line computation is reduced to feedback rule evaluation or table lookup interpolation. However, it is well known that this approach suffers from the *curse of dimensionality* and is intractable for realistically sized JAO problems.

A subtle but significant attribute of the SDP formulation is that it explicitly models the feedback mechanism, i.e. the dependence of future control decisions on future information arrival, in the mathematical formulation. By modeling the feedback mechanism in the mathematical formulation, the SDP formulation generations *proactive* versus *reactive* solutions that are able to anticipate likely contingencies; this trait produces guaranteed superior performance for stochastic problems [2]. This proactive attribute is imperative for stable and agile control of the JAO enterprise because future information arrival and control opportunities are dependent on stringent spatial, temporal, and coordination constraints.

Note, in this paper, we have adopted Dreyfus's [2] terminology with respect *proactive* versus *reactive* feedback control solutions. **Closed-Loop Feedback** (CLF) will refer to feedback control solutions that explicitly model the feedback mechanism, and **Open-Loop Feedback** (OLF) will refer to feedback control solutions that neglect the feedback mechanism in the control optimization problem.

Given the well-known strengths and weaknesses of the SDP formulation, there has been a great deal of research on ASDP methods in recent years. These methods gen-

J. M. Wohletz is a Senior Research Scientist at ALPHATECH Inc., 50 Mall road, Burlington, MA 01803 USA, Member IEEE (e-mail: jwohletz@alphatech.com)

D. A. Castañon is a Professor in the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA, Member IEEE (e-mail: dac@bu.edu)

M. L. Curry is a Research Engineer at ALPHATECH Inc., 50 Mall road, Burlington, MA 01803 USA, Member IEEE (e-mail: mikec@alphatech.com)

erally maintain the SDP structure, but use a variety of techniques to approximate the optimal cost-to-go. In this paper, we apply one such technique known as the Rollout Algorithm [3], [4] to the JAO problem. The rollout algorithm — which has been used for a wide variety of dynamic decision problems [4], [5], [6], [7] — is a technique that exploits knowledge of a suboptimal decision rule to obtain an approximate cost-to-go for use in the SDP framework. Because the RA maintains the SDP structure, it falls within the category of closed-loop control; accordingly, the terminology RA and closed-loop feedback will be used synonymously in this paper.

An overview of the RA is presented below. Consider a discrete event version of a dynamic decision problem,

$$x_{k+1} = f_k(x_k, u_k, w_k) \tag{1}$$

where x_k is the state taking values in some set X_k, u_k is the control to be selected from a finite set $U_k(x_k)$, w_k is a random disturbance, and f_k is a given function. We assume that the disturbance $w_k, k = 0, 1, ...$ has a given distribution that depends explicitly only on the current state and control. Define a control policy, which is a sequence of feedback functions $\mu_k(x_k)$ that map each state x_k to control a u_k:

$$\pi_k = \{\mu_k(x_k), \mu_{k+1}(x_{k+1}), ..., \mu_{k+N-1}(x_{k+N-1})\} \tag{2}$$

thus, the control at time k is $u_k = \mu_k(x_k) \in U_k(x_k)$. In the N-stage horizon problems considered herein, the single-stage cost function is denoted by $g_k(x_k, \mu_k(x_k), w_k)$ and the terminal cost function is denoted by $G_{k+N}(x_{k+N})$. The cost-to-go for policy π starting from state x_k at time k can be computed as follows:

$$J_k^\pi(x_k) = E\left\{G_{k+N}(x_{k+N}) + \sum_{i=k}^{k+N-1} g_i(x_i, \mu_i(x_i), w_i)\right\} \tag{3}$$

and can be represented in the SDP recursion format as follows

$$J_k^\pi(x_k) = E\left\{g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}^\pi(f(x_k, \mu_k(x_k), w_k))\right\} \tag{4}$$

for all k and with the initial condition

$$J_{k+N}^\pi = G_{k+N}(x_{k+N}) \tag{5}$$

The N-stage, SDP solution is as follows

$$\begin{aligned} \pi_k^* \quad = \quad & \arg\min_{\pi_k, u_i \in U_i(x_i)} E\left\{g_k(x_k, \mu_k(x_k), w_k)\right. \\ & \left. + J_{k+1}^\pi(f(x_k, \mu_k(x_k), w_k))\right\} \end{aligned} \tag{6}$$

The RA exploits this formulation by replacing the control mapping for times $k+1 \rightarrow k+N-1$ with a predetermined baseline heuristic $\bar{\mu}(x_i)$. Additionally, the RA is solved forward in time, and is computed at the actual state x_k versus all possible states at time k. Thus, the RA has the following policy:

$$\pi_k^{RO} = \{u_k(x_k), \bar{\mu}(x_{k+1}), ..., \bar{\mu}(x_{k+N-1})\} \tag{7}$$

Using this policy, the approximate optimal control solution at time k is

$$\begin{aligned} u_k^{RO} \quad = \quad & \arg\min_{u_k \in U_k(x_k)} E\left\{g_k(x_k, u_k, w_k)\right. \\ & \left. + J_{k+1}^{\pi^{RO}}(f(x_k, u_k, w_k))\right\} \end{aligned} \tag{8}$$

Thus, the rollout policy is a one-step lookahead policy with the optimal cost-to-go approximated by the cost-to-go of the base policy. The RA computes the best control at the current state x_k at time k by balancing the current cost with an approximate cost-to-go using a baseline heuristic to model future control decisions.

The computation of the cost-to-go of the base policy is by no means trivial. When the number of states is very large, the recursion may be infeasible. A straightforward approach for computing the rollout control at a given state x_k and time k is to use Monte Carlo simulations of the baseline policy. To implement this approach, we consider all possible controls $u_k \in U(x_k)$ and generate a "large" number of simulation trajectories starting from x_k. Thus, the simulated trajectory has the form:

$$\begin{aligned} x_{k+1} \quad &= \quad f(x_k, u_k, w_k) \\ x_{i+1} \quad &= \quad f(x_i, \bar{\mu}(x_i), w_i) \quad i = k+1, ..., k+N-1 \end{aligned} \tag{9}$$

The costs corresponding to these trajectories are averaged to obtain the Q-factor

$$Q(x_k, u_i) = E\left\{g_k(x_k, u_i, w_k) + J_{k+1}^{\pi^{RO}}(f(x_k, u_i, w_k))\right\} \tag{10}$$

Due to the finite number of simulations, only an estimate for the Q-factor $\hat{Q}(x_k, u_i)$ is obtained. The approximation becomes increasingly accurate as the number of simulation trajectories increases. Once the estimated Q-factor $\hat{Q}(x_k, u_i)$ corresponding to each candidate control $u_i \in U(x_k)$ is computed, the optimal rollout control at time k for state x_k is

$$\hat{u}_k^{RO} = \arg\min_{u_k \in U_k(x_k)} \hat{Q}(x_k, u_k) \tag{11}$$

Thus, the RA starts with a baseline heuristic and improves the policy by using on-line learning and simulation. The algorithm is related to the SDP formulation and is based on policy iteration ideas. As a result, the RA is not as ambitious as the SDP, and only provides modest guarantees of near-optimality [4]. It is an intermediate methodology between heuristics and SDP.

III. Simulation Results

This section presents implementation and simulation results of the RA applied to a simplified JAO environment. The purpose of this implementation is to highlight the benefits of approximate optimal control for the JAO enterprise. Thus, results will be presented for open-loop (OL), open-loop feedback (OLF), and closed-loop feedback (CLF) controllers, and both performance and behavioral characteristics will be highlighted. All controllers generate mission orders based on the current state of the JAO environment. A mission order includes target assignment, coarse level routing, air package composition, and desired time-on-target.

A. Demonstration Scenario

As a proof of concept, a small and simple scenario, illustrated in Figure 1, is used to demonstrate the benefits of CLF for the JAO enterprise. The scenario has a single airbase, located to the left, that includes three strike and three weasel aircraft. Enemy assets include two surface-to-air missiles sites (SAMs) and six targets of which T_2 is fleeting and T_4 is either fleeting —Scenario A— or time critical —Scenario B.

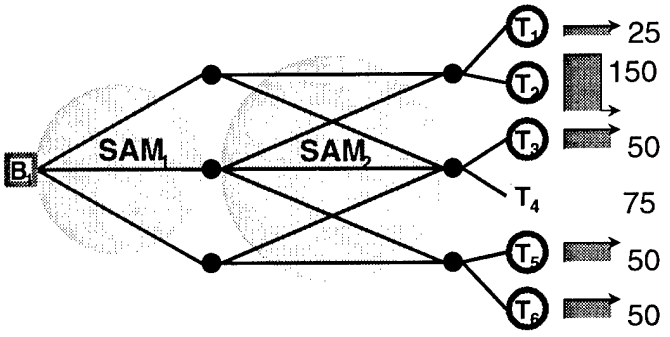


Fig. 1. Simplified JAO Vignette

The important attributes of this scenario include limited assets, risk/reward dependent on package composition, basic threat avoidance routing, and multiple targets, some of which are fleeting and emerging. Asset attrition is highly likely since SAMs occupy the airspace between the airbase and targets. Also, given the number of targets and limited assets, multiple strike packages and waves will be required to service the targets. Finally, the performance in this scenario is governed by the controllers' ability to manage attrition while servicing the fleeting targets.

The state dynamics in (1) can be represented as a discrete event system using a finite state, stochastic timed automaton formulation[9]. As will be shown, the state dynamics can be constructed through composition of individual asset automata. Accordingly, the individual asset dynamics, i.e. aircraft, threats, and target, will be presented followed by the composition of these dynamics into a high-level, stochastic timed automaton. To simplify this discussion, the notion of an automaton for each asset will be illustrated via a state transition diagram. The state transition diagram for an aircraft is illustrated in Figure 2. Based on this diagram, the aircraft

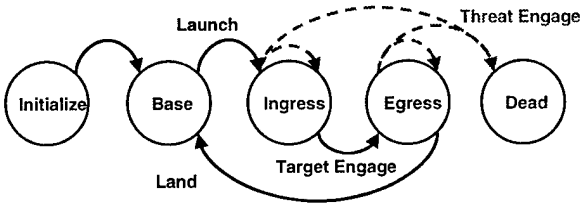


Fig. 2. Aircraft State Transition Diagram

state is defined as $\mathcal{X}_{AC} = \{Base, Ingress, Egress, Dead\}$ and significant event set is defined as $\mathcal{E}_{AC} = \{Launch, Threat\ Engage, Target\ Engage, Land\}$. As mention at the beginning of this section, the output of the control optimization problem presented in Section II is mission orders that consist of list of air packages assigned to targets. An air package AP_i is defined as the product composition of n strike aircraft AC_s and m weasel aircraft AC_w:

$$AP_i = AC_{s_1} \times \cdots \times AC_{s_n} \times AC_{w_1} \times \cdots \times AC_{w_m} \quad (12)$$

Accordingly, the composite state is defined as $\mathcal{X}_{AP_i} = \{\mathcal{X}_{AC_{s_1}}, \cdots, \mathcal{X}_{AC_{w_m}}\}$ and the event set is defined as $\mathcal{E}_{AP_i} = \{Launch, ThreatEngage, TargetEngage, Land\}$. The distinguishing feature between strike and weasel aircraft is that a strike aircraft destroy targets where weasel aircraft destroy SAMs. A SAM may destroy both aircraft types. As illustrated in the above diagram, all transitions are deterministic with the exception of the threat engagement. For the threat engagement event, the state transition $p(x_{AP_{k+1}}|x_{AP_k}, x_{SAM_k}, Threat\ Engage)$ is dependent on both air package state x_{AP} and the threat state x_{SAM}, and is thus defined as an interaction event.

The state transition diagram for a target is illustrated in Figure 3. Thus, the target state $\mathcal{X}_T =$

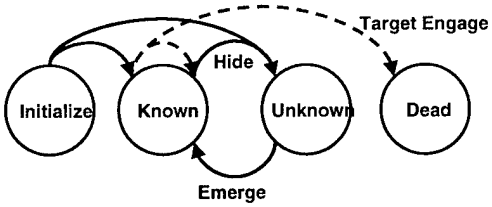


Fig. 3. Target State Transition Diagram

$\{Known, Unknown, Dead\}$ and significant events $\mathcal{E}_T = \{Emerge, Hide, Target\ Engage\}$. For the scenario presented in Figure 1, there are four normal targets, T_1, T_3, T_5, and T_6 that have the following constraints :$p_o(Known) = 1$ and neither $\{Hide\}$ or $\{Emerge\}$ is a triggering event $w.p.$ 0. Target T_2 is a fleeting target with the following constraints: $p_o(Known) = 1$ and $\{Emerge\}$ is a triggering event $w.p.$ 0. For Scenario A, target T_4 is fleeting and has identical transitions to target T_2. For Scenario B, target T_4 is a time critical target with the following constraints: $p_o(Unknown) = 1$. For all of these target types, the state can only transition to *Dead* from the *Known* state, and this interaction event is governed by transition matrix $p(x_{T_{k+1}}|x_{AP_k}, x_{T_k}, Target\ Engage)$.

Finally, the state transition diagram for the SAM is illustrated in Figure 4. Thus, the SAM state $\mathcal{X}_{SAM} = \{Active, Inactive, Dead\}$ and significant events $\mathcal{E}_{SAM} = \{Activate, Deactivate, Threat\ Engage\}$. It is important to note that the threat engagement event can only occur if the SAM is *Active*, i.e. radar on, and this transaction is governed by the transition matrix $p(x_{SAM_{k+1}}|x_{AP_k}, x_{SAM_k}, Threat\ Engage)$.

Thus, state dynamics (1) can be constructed through

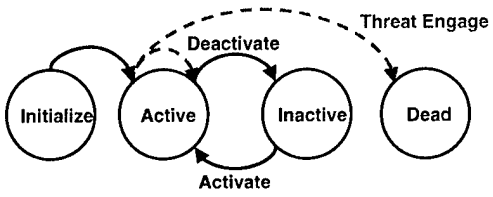


Fig. 4. SAM State Transition Diagram

parallel composition $\|$ of the individual asset automata:

$$\begin{aligned} x_{k+1} &= AP_1\| \cdots \|AP_n\|T_1\| \cdots \|T_6\|SAM_1\|SAM_2 \\ &= (\mathcal{X}, \mathcal{E}, \Gamma(x_k), p(x_{k+1}|x_k, e_{k+1}), p_o(x_o), G) \end{aligned} \tag{13}$$

where $\mathcal{X}$ is the composite state space for each asset, $\mathcal{E}$ is the composite event set, $\Gamma(x)$ is a state dependent set of feasible events, i.e. $\Gamma(x) \subseteq \mathcal{E}\ \forall\ x \in \mathcal{X}$, $p(x_{k+1}|x_k, e_{k+1}) = P\{\mathbf{x_{k+1}} = x_{k+1}|x_k, e_{k+1}\}$ is the state transition probability defined for all $x_{k+1}, x_k \in \mathcal{X}$, $e_{k+1} \in \mathcal{E}$, and such that $p(x_{k+1}|x_k, e_{k+1}) = 0\ \forall\ e_{k+1} \notin \Gamma(x_k)$, $p_o(x) = P\{\mathbf{x} = x_o\}\ \forall\ x \in \mathcal{X}$, and $G_{i,j,k}$ is the clock structure for event $i \in \mathcal{E}$, air package $j \in \{AP_1, \ldots, AP_n\}$, and enemy asset $k \in \{T_1, \ldots, T_6, SAM_1, SAM_2\}$. Note, the subscripts i or j may be omitted if the interaction is irrelevant.

The definition of the first five terms of the automaton in (13) follow from the discussion in the previous paragraphs, and all that remains is to define the stochastic clock structure $G_{i,j,k}$ that defines the triggering event. To simplify the composite clock structure, time k has been normalized into unit increments T where each increment corresponds to a transition from one waypoint to another in Figure 1. Thus, it take $2.5T$ to transition from the base to a target; likewise, it takes $6T$ to perform a cycle: $5T$ for a base-target-base transition and T to turn aircraft around. Another simplification is introduced by limiting the control loop closures, i.e. formation and launching new air packages, to time increments of $w3T$ where $w \in \mathbb{Z} \geq 0$ is the wave number.

Given these simplifications, the clock structure for a given air package AP_j launched at $k = 3wT$ is as follows:

$$\begin{aligned} G_{LU,PA_j} &= \{w3T\}\ w.p.\ 1 \\ G_{DE,PA_j,SAM_1} &= \{(w3+1)T, (w3+5)T\}\ w.p\ 0.5 \\ G_{TR,PA_j,SAM_1} &= \{(w3+1)T, (w3+5)T\}\ w.p.\ 1 \\ G_{AC,PA_j,SAM_1} &= \{(w3+2)T, (w3+6)T\}\ w.p\ 1 \\ G_{DE,PA_j,SAM_2} &= \{(w3+2)T, (w3+4)T\}\ w.p.\ 0.5 \\ G_{TR,PA_j,SAM_2} &= \{(w3+2)T, (w3+4)T\}\ w.p.\ 1 \\ G_{AC,PA_j,SAM_2} &= \{(w3+2)T, (w3+4)T\}\ w.p.\ 1 \\ G_{TA,PA_j,T_k} &= \{(w3+3)T\}\ w.p.\ 1 \\ G_{LN,PA_j} &= \{(w3+6)T\}\ w.p.\ 1 \end{aligned}$$

where $G_{LU,PA_j} = \{w3T\}\ w.p.\ 1$ implies $P\{PA_j\ Launch\ @\ k = w3T\} = 1$. Additionally, the clock sequences for targets T_2 and T_4, which are dependent on clock time cT where $c \in \mathbb{Z} > 0$, are as follows:

$$\begin{aligned} G_{HD,T_2} &= \{10T\}\ w.p.\ 1 \\ G_{HD,T_4} &= \{5T\}\ w.p.\ 1 \\ \text{or} & \\ G_{EM,T_4} &= P\{Emerge \leq cT\} = \textstyle\sum_i^c 0.2(1-0.2)^{i-1} \\ G_{HD,T_4} &= \{G_{EM,T_4} + 3T\}\ w.p.\ 1 \end{aligned}$$

where the distinction is made for Scenario A and B.

Due to this time synchronization and the fact that multiple air packages $AP_1\| \cdots \|AP_n$ maybe in a given wave, multiple events will occur at the same time. As a result, a priority rule is imposed such that all feasible non-interaction events, i.e. $\{Launch, Land, Emerge, Hide, Activate, Deactivate\}$, occur first followed by feasible interacting events, i.e. $\{Threat\ Engage, Target\ Engage\}$. In the case of multiple air packages engaging the same SAM, the engagements are treated separately and the triggering events are executed in the order of lowest to highest package number.

B. Controller Implementation

As noted in the previous section, control loop closures, i.e. optimization problem solved to determine formation and launching new air packages, occur on time intervals of $w3T$ where $w \in \mathbb{Z} \geq 0$ is the wave number. For this scenario, the objective of the control problem is to service as many targets possible while preserving our own force. To quantify this objective, a valuation scheme is used. Targets are valued, depicted in Figure 1, from 25 to 150, strike aircraft at 20, weasel aircraft at 40, and no values are assigned to the SAMs. Thus, the objective function may be represented as follows:

$$\begin{aligned} J_N^{\pi^{RO}}(x_N) &= E\Big\{\textstyle\sum_i^6 J_{T_i} 1(x_{T_i}(N) = Dead) \\ &+ \textstyle\sum_i^3 -20\ 1(x_{AC_{s_i}}(N) = Dead) \\ &+ \textstyle\sum_i^3 -40\ 1(x_{AC_{w_i}}(N) = Dead)\Big\} \end{aligned} \tag{14}$$

where $1(x_{AC_{s_i}}(N) = Dead)$ in an indicator function and equals 1 if $x_{AC_{s_i}}(N) = Dead$ and 0 otherwise, and $N = \infty$ is the planning horizon used for this implementation. As noted in Section II, the Q-factor estimate, $\hat{Q}(x_k, u_i)$ is substituted for (14), and the optimal rollout control $\hat{u}_k^{RO} \in U(x_k)$ is determined from (11). For this implementation, 10 simulations trials where used to compute $\hat{Q}(x_k, u_i)$. Also, the admissible control set $U(x_k)$ can be summarized as follows: aircraft at base can be sent out on missions to specific targets, for which the minimum risk route is determined *a priori*, or they can stay at base in reserve.

As illustrated in Section II, the RA uses a baseline heuristic $\bar{\mu}(x_i)$ to model future decisions. For this implementation, a generic greedy planning heuristic that generates a complete set of mission orders is used. The heuristic is greedy because for each target, the best mission package is selected without consideration to the available assets and the mission priority, which is proportional to the value of the target. The details of this algorithm are as follows: for each target T_i, the best mission packaged—consisting of n strike and m^* weasel aircraft—is determined via the following maximization:

$$\begin{aligned} (n^*, m^*) = \arg\max_{i,j \leq 2} &\big[J_{T_i} P\{x_{T_i}(k) = Dead\} \\ &-20 \textstyle\sum_i^n P\{x_{AC_{s_i}}(k) = Dead\} \\ &-40 \textstyle\sum_i^m P\{x_{AC_{w_i}}(k) = Dead\}\big] \\ &/\ [20n + 40m] \end{aligned} \tag{15}$$

the probabilities are computed by propagating a Markov chain from time $k = 3wT$ to $k = 3wT + 6T$ using the dynamics in (13) with only a single air package with n strike and m weasel aircraft Note, $k = 3wT + 6T$ corresponds to the end of the planning horizon which is a single wave launched at $3wT$.

A few properties of the greedy heuristic are noteworthy. First, as a simplification, the target's window of vulnerability is not modeled in the Markov chain; Couple this with the 1-wave lookahead, the heuristic is not capable of recognizing the importance of aggressively prosecuting fleeting and time critical targets. Additionally, resource constraints are not modeled. As a result, the greedy algorithm's output is a complete list of mission orders and should be viewed as a launch queue where only the highest priority missions are launched given the available resources.

C. Simulation Results

In this section, RA simulation results are presented for both versions of the JAO scenario described above. As a basis of comparison, these results will be presented relative to a *loose* Stochastic Upper Bound (SUB) and relative to both OL and OLF controller implementations. For both versions of the JAO scenario, a SUB is computed optimistically by determining the expected value $E\{J(x_\infty)\}$ (14) while assuming no threats, and is thus a *loose* bound.

The comparison of the RA with OL and OLF will quantify the performance benefits of feedback control over OL and the performance benefit of a *proactive* versus *reactive* control. For the OL and OLF controller implementations, the greedy heuristic described in Section III-B is used to generate mission orders. In the OL implementation, a list of mission orders for each target is determined based on the state of the environment at $k = 0$, and this plan is not updated. In the OLF implementation, the mission orders are updated for each loop closure time $k = w3T$ for $w \in \mathbb{Z} \geq 0$.

C.1 Scenario A Results

Figure 5 shows the Scenario A performance results of the OL, OLF, and CLF strategies. These results show that the RA outperforms the OLF strategy, which attains a statistically significant improvement over the corresponding OL strategy. It is seen that the optimal control framework was able to achieve 86% of a *loose* SUB whereas the OLF was only able to achieve 67% of the bound, and the OL controller was only able to achieve 45% of the bound. The performance improvement is attributed to the fact that the RA develops strategies that are not inherent in the baseline heuristic. For Scenario A, these strategies include staging packages and opening attack corridors to manage asset attrition, aggressively prosecuting fleeting targets, and reserving assets for likely contingencies. Thus, a simple, generic heuristic used in the rollout framework does generate near-optimal behaviors and results.

As an example, at $k = 0$ when all targets and SAMS are alive, the RA sends a $AP_1[0,2] \rightarrow T_4$ followed by a $AP_2[2,1] \rightarrow T_4$ and leaves the remaining strike at base. Note, $AP_i[n,m] \rightarrow T_j$ is shorthand for AP_i with n strike

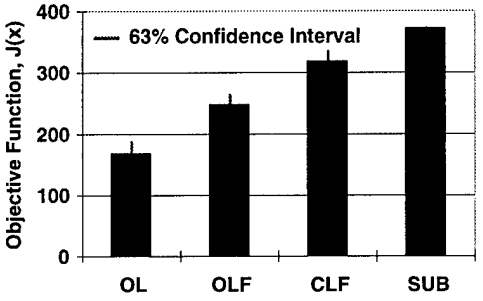


Fig. 5. Open-Loop, Open-Loop Feedback, and Closed-Loop Feedback Results for Scenario A (86 Monte-Carlo)

and m weasels assigned to T_j. For this same situation, the greedy heuristic recommends sending a $AP_1[2,2] \rightarrow T_2$ instead of T_4 and leaving the remaining strike and weasel aircraft at base since alternative missions appear too risky. This decision encapsulates many of the behavioral characteristics that are *learned* by rolling out candidate controls. The RA sends the $AP_1[0,2] \rightarrow T_4$ out first to manage attrition by opening an attack corridor. This mission is followed by a $AP_2[2,1] \rightarrow T_4$, which is time critical; a weasel aircraft is included in this package in the event that a SAM is alive. Furthermore, by modeling the next loop closure at $k = 3T$, a strike aircraft is held in reserve for the potential opportunity to launch at T_2 in the event that both SAMs are destroyed. This provides a maximum of two strike opportunities on this high valued target. Thus, the RA *learns* the concept of time and managing attrition without coaching since neither of these traits is modeled in the baseline heuristic. Granted, a more sophisticated baseline heuristic could have been used; however, by customizing the baseline heuristic, generality would be lost which is clearly undesirable for applicability to other JAO scenarios. But the point must be stressed, that by modeling the feedback mechanism, the RA was able to position assets for opportunities of recourse, and this resulted in better performance over the OLF implementation.

C.2 Scenario B Results

Figure 6 shows the Scenario B performance results of the OL, OLF, and CLF strategies. Again, these results show that RA outperforms the OLF strategy, which attains a statistically significant performance improvement over the corresponding OL strategy. It is seen that the optimal control framework was able to achieve 81% of a *loose* SUB whereas the OLF was only able to achieve 73% of the bound and the OL controller was only able to achieve 52% of the bound. Like Scenario A, the performance improvement is attributed to the fact that the RA develops strategies that are not inherent in the baseline heuristic. For Scenario B, these strategies include developing a Combat Air Patrol (CAP) over the emerging target region, staging packages and opening attack corridors to manage asset attrition, aggressively prosecuting fleeting targets, and reserving assets for likely contingencies.

Of these behaviors, the CAP is the most interesting. Be-

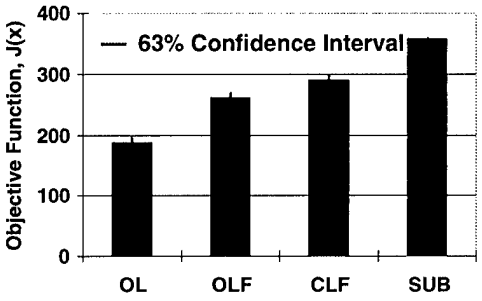


Fig. 6. Open-Loop, Open-Loop Feedback, and Closed-Loop Feedback Results for Scenario B (200 Monte-Carlo)

cause of the uncertain value function for T_4, the RA develops a policy of continuously maintaining an ingress mission to T_4 as soon as possible. Note, this simulation environment does not have loiter capability. At $k = 0$ when all targets and SAMs are alive, the RA sends a $AP_1[0,1] \rightarrow T_2$ followed by a $AP_2[2,2] \rightarrow T_2$ and leaves the remaining strike at base. The greedy heuristic's recommendation is the same as it was for Scenario A. The RA sends the AP_1 out first to manage attrition by opening an attack corridor. The mission is then followed by a AP_2 to aggressively prosecute T_2 so that it can free up resources to establish the CAP. Given that T_4 emerges on average at $\approx 4T$, the reserve aircraft will be in position to launch at the next loop closure $k = 3T$. Likewise, when the first wave returns to base, these aircraft will immediately be turned around to continue the CAP. This process will continue until the target emerges and expires. The remaining normal targets are then serviced.

Viewing the results presented in Figure 5 and Figure 6, it appears that OL and OLF performance increases for Scenario B. It is true that there is a performance percentage increase; however, this is a result of the invariance of the OL and OLF results and a decrease in the SUB between the two versions. The OL and OLF results are statistically equivalent because neither control strategy services T_4 regardless whether the value function is deterministic or stochastic. On the other hand, the decrease in the SUB and the resulting reduction in rollout performance reflect the difficulty of Scenario B. Because of the requirement to maintain an ingress mission to T_4 in anticipation of the target emerging, fewer aircraft, on average, are able service the high-value, fleeting target T_2.

IV. Conclusions

This paper focuses on the problem of providing a military commander with real-time, closed-loop feedback control of Joint Air Operations (JAO) via near-optimal mission assignments, which anticipate possible mission modifications due to uncertain future events. Based on the theory of stochastic dynamic programming, an approximate optimal control strategy known as the rollout algorithm is presented in this paper. In this framework, the feedback mechanism, i.e. the dependence of future control decisions on future information arrival, is explicitly modeled in the control optimization problem; as a result, future significant events are anticipated and assets are hedged for opportunities of recourse. Thus, the resulting missions can be adapted to contingencies with minimal performance degradation, resulting in robust, stable control. The rollout algorithm is applied to a small JAO scenario that includes limited assets, risk/reward that is dependent on package composition, basic threat avoidance routing, and multiple targets, some of which are fleeting and emerging. Simulation results illustrate the benefits of the approximate optimal control strategy. It is shown that the rollout algorithm provides statistically significant performance improvements over an open-loop feedback strategy that uses the same baseline heuristic. The performance improvements are attributed to the fact that the rollout algorithm is able to learn near-optimal behaviors — establishing combat air patrol over time critical areas, staging packages and opening attack corridors to manage friendly asset attrition, aggressively prosecuting fleeting targets, and reserving assets for contingencies — that are not modeled in the baseline heuristic.

Acknowledgments

This research was supported by the Defense Advanced Research Projects, Information Systems Office (DARPA/ISO) and the Air Force Research Laboratory, Information Directorate, System Concepts & Applications Branch (AFRL/IFSA) under contract number F30602-99-C-0203.

References

[1] Bertsekas, D. P., *Dynamics Programming and Optimal Control, Vol I-II*, Athena Scientific, Belmont, MA, 1995.

[2] Dreyfus, S. E., "Some Types of Optimal Control of Stochastic Systems," *Journal SIAM Control*, Series A, Vol. 2, No. 1, pp. 120-134, 1962.

[3] Bertsekas, D.P., 'Rollout Algorithms: An Overview' *Proceedings of the 38th Conference on Decision & Control*, Phoenix, Arizona, December, 1999.

[4] Bertsekas, D.P., Tsitsiklis, J.N., and Wu, C., 'Rollout Algorithms for Combinatorial Optimization' *Journal of Heuristics*, Vol. 3., Num. 3, November, 1997, pp. 245-262.

[5] Bertsekas, D.P., Castañon, D.A., 'Rollout Algorithms for Stochastic Scheduling Problems' *Journal of Heuristics*, Vol. 5, 1999.

[6] Patek, S.D., Logan, D.A., and Castañon, D.A., 'Approximate Dynamic Programming for the Solution of Multiplatform Path Planning Problems' *IEEE Systems, Man, and Cybernetics 1999 Proceedings*, Vol. 1, 1999, pp. 1061-1066.

[7] Bertsekas, D.P., Castañon, D.A., Curry, M.L., and Logan, D.A., 'Adaptive Multi-platform Scheduling in a Risky Environment' *Advances in Enterprise Control Proceedings*, Symposium Sponsored by JFACC Program, DARPA/ISO, San Diego, CA, November, 15-16, 1999, pp. 121-127.

[8] Bertsekas, D.P., Castañon, D.A., Curry, M.L., Logan, D.A., and Wu, C., 'Dynamic Programming Methods for Adaptive Multiplatform Scheduling in a Risky Environment' *Advances in Enterprise Control Proceedings*, Symposium Sponsored by JFACC Program, DARPA/ISO, Minneapolis, MN, July, 10-11, 2000, pp. 121-128.

[9] Cassandras, C.G., and Lafortune, S., *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, MA, 1999.

Modeling and Agile Control for a Joint Air Operation Environment

C.G. Cassandras[a], K. Gokbayrak[a], D.A. Castanon[b,c]
J.M. Wohletz[c], M.L. Curry[c], and M. Gates[c]

[a]Dept. of Manufacturing Engineering, Boston University, Boston, MA 02215, USA

[b]Dept. of Electrical and Computer Engineering, Boston University, Boston, MA 02215, USA

[c]ALPHATECH, Inc., Burlington, MA 01803, USA

ABSTRACT

A key component of a Joint Air Operation (JAO) environment is the planning and dynamic control of missions in the presence of uncertainties. This involves the assignment of resources (e.g., different aircraft types) to targets while taking into account and anticipating the effect of random future events and, subsequently, dynamic control in response to various controllable and uncontrollable events as missions are executed in a hostile and rapidly changing setting. The objective is to maximize the reward associated with targets while minimizing loss of resources. In this paper, we first formulate the problem of optimal mission assignment and identify the complexities involved due to combinatorial and stochastic characteristics. We then describe a discrete event simulation tool developed to model the JAO environment and all of its dynamics and stochastic elements and to provide a testbed for several methods we are developing to solve the problem of agile mission control. We describe some of these methods, including approximate dynamic programming using rollout algorithms and optimal resource allocation schemes, and present some numerical results.

Keywords: Discrete Event System, Stochastic Dynamic Programming, Simulation, Mission Planning.

1. INTRODUCTION

The Joint Air Operations (JAO) environment may be viewed as a stochastic dynamic system in which "entities" such as aircraft, threats (e.g., hostile air defenses), and targets interact, and a variety of events take place, some of which are controlled (e.g., the decision of an aircraft to engage a target) and some are random (e.g., an aircraft being destroyed by a threat). Traditionally, operations are carried out based on a predefined plan, typically created as an Air Tasking Order (ATO) about 24 hours in advance of an actual mission. This approach, however, lacks "agility", since it cannot anticipate changes in the battle space nor take advantage of ever-increasing sensor capabilities that can provide additional information on a continuous basis. Therefore, a critical need is to develop dynamic control mechanisms that not only incorporate anticipative capabilities regarding future uncertain events, but are also able to swiftly react to observed events and make adjustments (e.g., retasking an airborne aircraft or aborting a mission).

In order to accomplish the goal of agile control, two essential tasks need to be carried out. First, we need to develop an appropriate modeling framework for the JAO environment. Since this environment is extremely complex, one is tempted to develop a highly detailed model, normally in a simulation setting. While such a model may contain great *descriptive* value, it is often of little use for *prescriptive* purposes, i.e., as the basis for deriving the agile control schemes we seek. This is because higly detailed models are not only computationally intensive so that real-time applications are out of the question, but they often obscure the salient features which are needed to determine the right action for a given situation (the analogy of "missing the forest for the trees" applies here). Therefore, the challenge is to identify the appropriate level of modeling which will yield "just enough" useful information to make optimal decisions. The second task is that of formulating and solving optimization problems, based on an appropriate battle space model, which capture the fundamental objective of JAO: maximize the reward associated with targets while minimizing the loss of friendly assets. While the formulation of such problems can be quite straightforward when an appropriate model is available, their solution is far from feasible due to a variety of complexities which we will discuss.

In this paper, we present a stochastic dynamic model for the battle space, which is designed to suit the JAO level of detail. This is accomplished by adopting a Discrete Event System (DES) framework[5] and identifying events and

state transitions which capture the key features of the processes describing battle space entity interactions (Section 2). This framework forms the basis of a simulation tool we have developed, which is briefly described in Section 3. In Section 4, we formulate a stochastic dynamic optimization problem the solution of which provides the agile control mechanism desired. The complexity of such a problem, however, is such that obtaining an exact solution is infeasible. One must therefore seek approximate solution methodologies. Some such methodologies are presented in Wohletz et al.,[14] where the advantages of closed-loop (dynamic) control schemes over open-loop (static) schemes are illustrated. In this paper, we include a more recent approach that combines a rollout strategy, as described in Wohletz et al.,[14] with the "surrogate problem" method presented in Gokbayrak and Cassandras.[8]

2. MODELING FRAMEWORK

We adopt a view of the battle space as consisting of several interacting entities:

1. Friendly *assets* which originate at bases. These assets are further classified into different types, such as Strike Aircraft (SA), Wild Weasels (WW), and Jammers. SA carry munition to be delivered to targets. WW provide support to the SA to protect them from enemy threats, while Jammers provide electronic air defense suppression capabilities.
2. *Threats* are typically air defense resources, such as SAMs, which may engage aircraft on their way to various targets or may be positioned so as to directly protect a target.
3. *Targets* are the ultimate destination of SA and they may contain their own air defense resources that can engage friendly assets.

A *mission* is the process of creating a "package" of friendly assets located at a base and assigning it to a target. The package is subsequently also assigned a *route* from base to target. For the purpose of JAO, we do not care to model the flight of each member of the package in detail. Instead, we represent the route from base to target as a set of "way points" $\{W_0, W_1, \ldots, W_n\}$, where W_0 denotes the base and W_n denotes the target. A way point is selected to represent a predefined point along the route or a known threat. In addition, there may be unknown threats in between way points, in which case the model must automatically create a new way point for the mission. The travel time between way points is a random variable based on aircraft speed and location of way points. Thus, a typical mission consists of traveling from one way point to the next and possibly engaging threats that may either be known or unknown. When (and if) the package arrives at its target, it engages it and subsequently must return to base by reversing its route (which may be modified depending on new information).

The *state* of a mission is defined by the number of each aircraft type in the package that are still alive and the location of the package. In addition, the state may include remaining munitions for each aircraft and other information that we shall not take explicitly into account at this level of modeling. The state of a target is binary representing whether it is "alive" or "destroyed". The state of a threat is similarly defined. Thus, the overall state of the battle space is described by the states of all missions, threats, and targets.

In order to systematically capture the dynamics of the battle space as its entities interact, as well as the effect of uncertain factors, we view it at the level of a DES and adopt a Stochastic Timed Automaton model.[5] A simple automaton is defined by (X, E, Γ, f, x_0), where (i) X is a countable set of states, (ii) E is a countable set of events, (iii) $\Gamma(x)$ is the set of feasible events when the state is x; this is a subset of E containing all events which are allowed to occur at state x, (iv) $f(x, e)$ is a state transition function such that when event e occurs at state x the next state is $x' = f(x, e)$; this can easily be replaced by a probabilistic mechanism such that the next state x' is determined with probability $p(x'; x, e)$, (v) x_0 is a given initial state. To obtain a *timed* automaton, a clock mechanism is added so that the stochastic DES can determine the next event to occur at state x, since $\Gamma(x)$ generally contains more than one feasible events. Thus, whenever an event occurs and the state is x, every event $i \in \Gamma(x)$ is associated with a clock value y_i and the next event to occur, denoted by e', is the one with the smallest clock value; formally:

$$e' = \arg \min_{i \in \Gamma(x)} \{y_i\}$$

When such an event occurs, the next state is simply given by

$$x' = f(x, e') \tag{1}$$

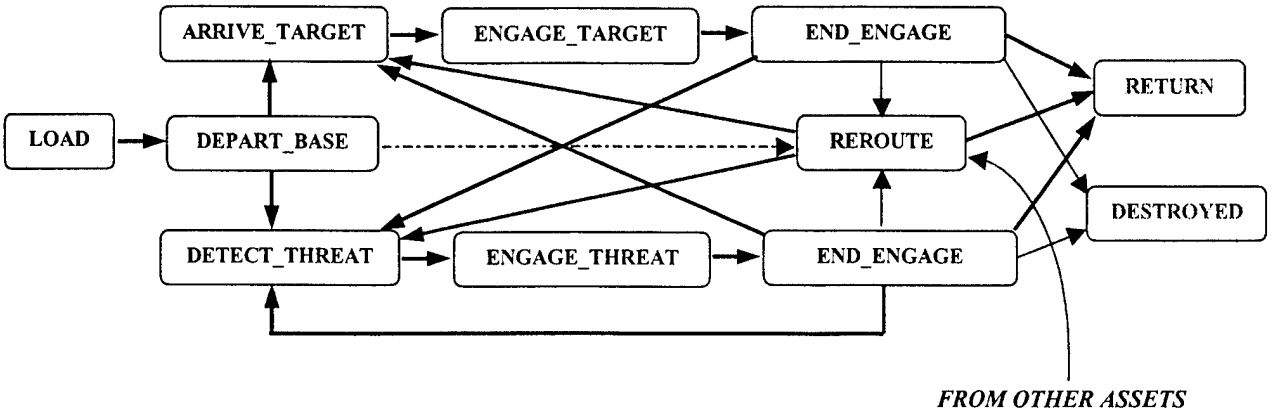


Figure 1. Mission related events

Moreover, if state x was entered at time t, then the new system time after event e' occurs is given by

$$t' = t + \min_{i \in \Gamma(x)} \{y_i\}$$

Observe that time is only updated when an event occurs; all battle space activity in between events is irrelevant to this model, a fact that maintains simplicity and computational efficiency. At time t' all events that remain feasible in the new state have their clocks decremented by setting

$$y_i' = y_i - \min_{i \in \Gamma(x)} \{y_i\}$$

since $\min_{i \in \Gamma(x)}\{y_i\}$ time units have elapsed since the last event. An event such that $i \in \Gamma(x)$ and $i \notin \Gamma(x')$ is eliminated and its clock discarded. Finally, if $e' \in \Gamma(x')$, this event is assigned a new clock value v_i (also referred to as the event's *lifetime*). In a *stochastic* timed automaton, this value is a random variable characterized by some distribution $G_i(\cdot)$. In a simulation setting, v_i is a sample from $G_i(\cdot)$ obtained through a pseudo-random number generator. Further details are omitted but may be found in Cassandras and Lafortune.[5]

The event set E required to model the battle space includes all events that may cause a state transition in any of the entities we have defined. The key events that describe a typical mission state evolution are shown in Figure 1 in the sequence in which they may occur. The mission is initiated at a base with a LOAD event, representing the process of loading munitions and configuring all assets to be included in the mission package. When this is completed, a DEPART_BASE event becomes feasible and may be scheduled. After this occurs, the package is physically airborne, on its way to the next way point in its route. Omitting way points that involve no threat encounters, the next event shown is either DETECT_THREAT or ARRIVE_TARGET. In the former case, a threat engagement process is initiated, whose details we omit. After END_ENGAGE occurs, the states of both the threat and the package will generally change. If no engagement actually takes place (e.g., because of the effect of jamming), then END_ENGAGE simply coincides with DETECT_THREAT. Note that a new DETECT_THREAT may occur, since subsequent way points can include additional threat encounters. It is also possible the package is destroyed or decides to abort the mission and return to base, as shown in Figure 1. In the case where the package reaches the target, a similar process takes place.

In addition, it is possible that the package makes a rerouting decision, which may consist of aborting the mission, selecting a new target, or specifying a new way point that includes a threat encounter. It is at this point that control decisions are critical and the role of feedback and solving an optimization problem enter the whole process. Moreover, note that these decisions may well depend on information supplied by other packages, which creates a "cooperative control" setting. In some cases, for example, it is possible to combine two or more packages and define a new mission or split a package up so that other missions are provided with additional assets. Although details are omitted, all these possibilities are part of this DES setting.

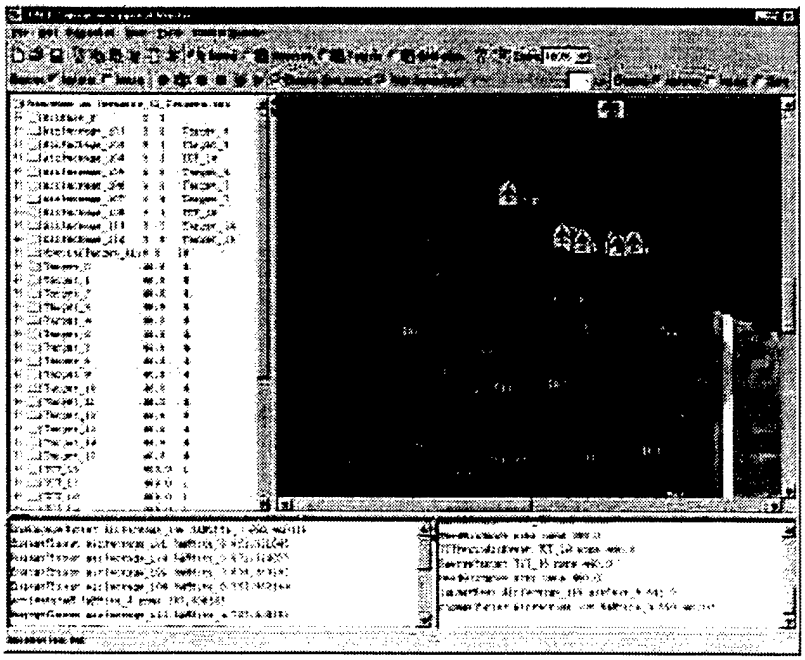


Figure 2. Typical screen snapshot of simulation tool

3. SIMULATION MODEL

Using the Stochastic Timed Automaton structure described in the previous section, a discrete-event simulation model was developed that includes a graphical user interface for defining the battle space and for monitoring all mission activity. A typical screen shot is shown in Figure 2, with a magnified view provided through a zooming in and out capability shown in Figure 3. The simulation environment allows one to define all battle space entities and their attributes (e.g., target location and value, threat location and range). When a simulation is executed, a state trajectory unfolds which may be graphically observed, while a detailed event trace is also provided (see bottom of screen in Figure 2).

The main value of such a simulation tool is to allow us to test different mission planning and control strategies. In addition, the underlying simulation engine may also be used as an integral part of some controllers which involve estimating alternative performance-related quantities.

4. OPTIMAL MISSION PLANNING AND AGILE CONTROL

The dynamics of the battle space, viewed as a DES, are described by the state transition mechanism of the stochastic timed automaton previously described. Thus, letting $k = 1, 2, \ldots$ index all events that take place over a given time interval, we can rewrite the state transition function (1) as

$$x_{k+1} = f(x_k, u_k, w_k) \tag{2}$$

where $x_k \in X$ is the battle space state, u_k is a control decision which can be made following the kth event and which is selected from a finite set $U_k(x_k)$, and w_k represents random factors that affect the state. In general, the control decision u_k is a function of the observed state x_k, so we can write $u_k = \mu_k(x_k) \in U_k(x_k)$. A *control policy* π_k applied for a time horizon that includes N events into the future is a sequence of functions that map each state x_i to a control u_i for all events $i = k, \ldots, k + N - 1$:

$$\pi_k = \{\mu_k(x_k), \mu_{k+1}(x_{k+1}), ..., \mu_{k+N-1}(x_{k+N-1})\} \tag{3}$$

In order to make optimal choices whenever a decision is made, cost functions $g_k(x_k, \mu_k(x_k), w_k)$ for all $k = 1, 2, \ldots$ are defined that quantify the immediate effect of selecting $u_k = \mu_k(x_k)$. In addition, a terminal cost function $G_{k+N}(x_{k+N})$

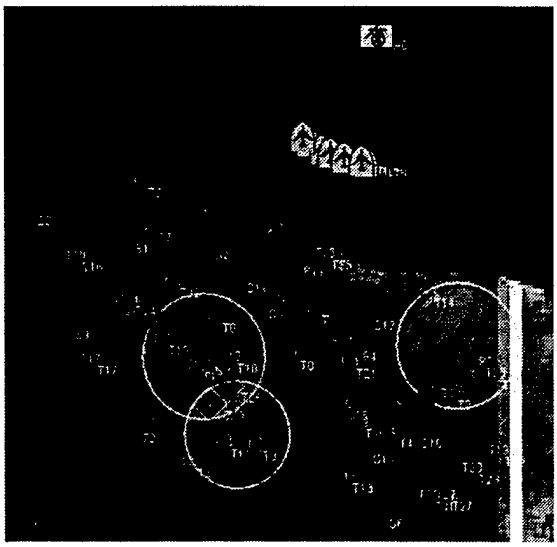


Figure 3. Magnified screen snapshot of simulation tool

is defined. Then, the total expected cost incurred by a policy initiated at k (also known as the expected cost-to-go) is

$$J_k^\pi(x_k) = E\left\{G_{k+N}(x_{k+N}) + \sum_{i=k}^{k+N-1} g_i(x_i, \mu_i(x_i), w_i)\right\} \tag{4}$$

which can also be rewritten recursively as

$$J_k^\pi(x_k) = E\left\{g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}^\pi(f(x_k, \mu_k(x_k), w_k))\right\}, \quad k = 1, 2, \ldots \tag{5}$$

with the initial condition

$$J_{k+N}^\pi = G_{k+N}(x_{k+N}) \tag{6}$$

This is the basis of the well-known Stochastic Dynamic Programming (SDP) algorithm[2] which provides a solution to the problem of determining $\pi_k^* = \{u_k^*, ..., u_{k+N-1}^*\}$ minimizing the total expected cost; in particular, the optimal policy is obtained from

$$u_k^* = \arg \min_{u_k \in U_k(x_k)} E\left[g_k(x_k, u_k, w_k) + J_{k+1}^*(f(x_k, u_k, w_k))\right] \tag{7}$$

where $J_{k+1}^*(f(x_k, u_k, w_k))$ is the optimal cost-to-go starting at $k+1$ with $J_{k+N}^* = G_{k+N}(x_{k+N})$. Although in principle one can apply this algorithm to obtain an explicit solution to the problem, this approach is computationally intractable for all but very simple problems. When it comes to using it in the JAO context, there are four broad areas of complexity we are facing: *combinatorial*, *stochastic*, *distributed*, and *computational*. First, the problem is combinatorially complex since the number of states and control options grows exponentially with the number of assets and targets. Second, the time scale of interest in mission control is long, which results in a high degree of future uncertainty contributed by (among many factors) sensor inaccuracies, unexpected hostile actions, and deviations from a plan upon its execution. Another difficulty arises from distributed complexity, which includes collection and management of information from different assets, together with coordination and communication strategies to achieve cooperative control. Finally, computational complexity is due to the need for real-time decisions which must normally be implementable onboard an aircraft.

One of the most common ways used to overcome these complexities is based on "decomposition" of the overall problem into smaller, more manageable components. If such decomposition is time-based, we can first solve an optimization problem aimed at determining $u_k = \mu_k(x_k)$ that minimizes

$$J_k^u(x_k) = E\left\{\sum_{i=k}^{k+N-1} g_i(x_i, u_k, w_i)\right\} \tag{8}$$

instead of (4) over a limited time horizon reflected by the choice of N. Since there are often events following which control actions may not be taken, it is reasonable to decompose a campaign in this fashion, choosing appropriate time intervals based on "significant events". For example, when packages return to base it is reasonable to collect all returning assets and plan new missions; on the other hand, an event such as detecting a threat may not warrant re-evaluation of all possible control actions. Although (8) is a static optimization problem requiring the specification of a single control u_k, rather than a sequence $\{u_k, u_{k+1}, ..., u_{k+N-1}\}$, it is still a very challenging task. The drawback of this approach is that it prevents the controller from making decisions that take into account future events beyond the selected time horizon. As an example, if a time-critical high-value target is identified for some future time outside the present decision horizon, the controller is not able to reserve adequate resources to define a mission for this target. To deal with this issue, there are two options. First, one can extend the time horizon and increase the dimensionality of the the control vector u_k to account for future decisions, thus trading off performance for computational efficiency. Alternatively, one can approximate the expected cost-to-go by replacing $\mu_i(x_i)$ in (4) for $i > k$ with a baseline heuristic $\bar{\mu}(x_i)$. At each subsequent event, the problem can be resolved "rolling out" the time horizon forward, hence the term "rollout algorithm"[3] which was used in Wohletz et al.[14] In this case, the approximate optimal control obtained at k is

$$u_k^{RO} = \arg \min_{u_k \in U_k(x_k)} E\left[g_k(x_k, u_k, w_k) + J_{k+1}^{\pi^{RO}}(f(x_k, u_k, w_k))\right] \tag{9}$$

Thus, the rollout algorithm seeks the best control at state x_k that includes the current one-step cost and an approximate cost-to-go obtained through a baseline heuristic to model future decisions. However, evaluating the expected value $E[J_{k+1}^{\pi^{RO}}(f(x_k, u_k, w_k))]$ is still a difficult problem. More generally, the problem is to estimate the so-called Q-function

$$Q(x_k, u_i) = E\left\{g_k(x_k, u_i, w_k) + J_{k+1}^{\pi^{RO}}(f(x_k, u_i, w_k))\right\} \tag{10}$$

for any control $u_i \in U(x_k)$. One approcah is to estimate this expectation through simulation. Thus, using the simulation model discussed earlier we can generate sample paths starting at state x_k and satisfying

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, w_k) \\ x_{i+1} &= f(x_i, \bar{\mu}(x_i), w_i) \quad i = k+1, ..., k+N-1 \end{aligned} \tag{11}$$

The resulting estimate is denoted by $\hat{Q}(x_k, u_i)$. Thus, this approach reduces to solving an optimization problem to determine

$$\hat{u}_k^{RO} = \arg \min_{u_k \in U_k(x_k)} \hat{Q}(x_k, u_k) \tag{12}$$

This problem is far from simple, as the dimensionality of $U_k(x_k)$ is typically enormous (in the billions of possible control choices). In the remainder of this paper, we will discuss an approach for its solution, based on the premise that the cost function $\hat{Q}(x_k, u_k)$ can be adequately approximated.

4.1. Optimal Mission Planning

We begin by using the model discussed in Section 2 for a JAO setting in order to identify the precise structure of the control choice set $U_k(x_k)$ when the state is x_k. Let us assume that this state reflects an initial condition whereby missions are being assigned to various targets at some base. Let M be the number of targets this base is responsible for and let Q be the number of different asset types that may be used to design strike packages (for our purposes, we set $Q = 3$ representing the three types of aircraft configurations mentioned earlier, i.e., SA, WW, and Jammers). The control decision can be expressed as a vector of dimensionality $M \cdot Q$ of the form

$$u = [u_{1,1}, \ldots, u_{1,Q}, \cdots, u_{M,1}, \ldots, u_{M,Q}] \tag{13}$$

where $u_{i,q}$ is the number of assets of type q allocated to target i. The set of possible decisions at this state, denoted by U, is limited by the capacity constraints

$$\sum_{i=1}^{M} u_{i,q} \leq K_q, \quad q = 1, \ldots, Q \tag{14}$$

where K_q is the total number of assets of type q available at the base. There may also be additional constraints such as $\beta_{i,q} \leq u_{i,q} \leq \gamma_{i,q}$ for $i = 1, \ldots, M$, if a package assigned to target i is not allowed to exceed $\gamma_{i,q}$ assets of type q

and is required to include at least $\beta_{i,q}$ assets. For example, a constraint may be imposed that no mission can use more than a given number of WW or that it must include at least one SA.

As already mentioned, the optimization problem we are interested in formulating is intended to maximize the reward obtained from successful destruction of targets while minimizing the cost of asset loss. In order to formulate such a problem, we associate a value V_i with the ith target and a cost C_q with an asset of type q. In addition, let $P_i^T(u)$ denote the probability of successfully destroying target i under a control vector u, and $P_i^q(u)$ the probability that an asset of type q is lost during the execution of the ith mission under u. Then, ignoring any future missions, the optimization problem is to determine u in (13) so as to maximize the total expected reward of the mission

$$J(u) = \sum_{i=1}^{M} \left[V_i P_i^T(u) - \sum_{q=1}^{Q} C_q u_{i,q} P_i^q(u) \right] \tag{15}$$

subject to the constraint (14) and possibly more mission-dependent constraints. Note that for some choices of u it is possible that $J(u) < 0$. Clearly, the solution of this problem requires knowledge of the probabilities $P_i^T(u)$ and $P_i^q(u)$ for $i = 1, \ldots, M$, $q = 1 \ldots, Q$ and for all possible $u \in U$. These probabilities depend on factors such as the outcome of engagements with threats and targets and the possible cooperation across missions that may fly over common threats, as well as basic parameters such as the firing rate of aircraft and the effectiveness of various weapons. It is possible to evaluate $P_i^T(u)$ and $P_i^q(u)$ analytically using the stochastic timed automaton model of Section 2 enhanced by more detailed engagement models and by making some simplifying assumptions. Alternatively, it is possible to estimate them through simulation, although this becomes a prohibitively time-consuming task. Yet another approach is not to attempt to solve an explicit optimization problem, but rather to rely on heuristics such as the "greedy heuristic" described in Wohletz et al.[14]

Note that the solution of (15) does not include any future decisions beyond a single wave of missions. Looking at (10), this problem considers only the first term and not the expected cost-to-go term $E[J_{k+1}^{\pi^{RO}}(f(x_k, u_k, w_k))]$. However, using a rollout approach as described earlier, this can be combined with (15) so that the problem we face is (12) with the control decision being of the form (13).

In what follows, we will present a methodology based on the "surrogate problem" idea[8] and provide some numerical examples comparing it to some alternatives.

4.2. The "Surrogate Problem" Method

A crucial difficulty with the problem of maximizing $J(u)$ in (15) is that the control vector u in (13) is discrete. This prevents us from using optimization techniques from conventional nonlinear programming with continuous decision variables, which are typically based on gradient information. The alternative is to rely on methods for discrete optimization. However, even in a deterministic setting this class of problems is NP-hard and one must rely on some form of a search algorithm (e.g., Simulated Annealing,[1] Genetic Algorithms[11]). In a stochastic environment such as in JAO, the problem is further complicated by the need to estimate the objective function of interest, such as $\hat{Q}(x_k, u_k)$ in (12). This generally requires Monte Carlo simulation or direct measurements made on the actual system. Most known approaches are based on some form of random search, as in algorithms proposed by Yan and Mukai,[15] Gong et al,[9] Shi and Olafsson.[13] Another recent contribution to this area involves the ordinal optimization approach presented in Ho et al[10] and used by Cassandras et al[4] to solve a class of resource allocation problems. The main difficulty with all these methods is that they are more suited to be off-line approaches lacking the real-time speed required in JAO.

The key idea in the "surrogate problem" method introduced by Gokbayrak and Cassandras[8] and generalized by the same authors[7] is to transform the *discrete* optimization problem (12) into a "surrogate" *continuous* optimization problem which is solved using standard gradient-based methods; its solution is then transformed back into a solution of the original problem. Thus, suppose that we begin by relaxing the integer constraint on all $u_{i,q}$ in (15) so that they can be regarded as continuous (real-valued) variables. The resulting "surrogate" problem then becomes: Find $\rho^* \in U_c$ that minimizes the cost function $J_c(\rho)$ over the continuous set U_c, i.e.,

$$J_c(\rho^*) = \min_{\rho \in U_c} J_c(\rho) = \min_{\rho \in U_c} E_\omega[L_c(\rho, \omega)] \tag{16}$$

where $\rho = [\rho_{1,1}, \ldots, \rho_{1,Q}, \cdots, \rho_{M,1}, \ldots, \rho_{M,Q}]$, $\rho_{i,q} \in \mathbb{R}_+$, is a real-valued vector, U_c is a constraint set such that $U \subset U_c$, and $L_c(\rho, \omega)$ is the cost function over a specific sample path (denoted by ω) when the state is ρ. Omitting

details which may be found in Gokbayrak and Cassandras[8],[7] we outline below the basic "surrogate problem" scheme. Initially, we set the surrogate control vector to be that of the actual one, i.e., $\rho_0 = u_0$. To avoid dealing with integer values in the surrogate problem, let us perturb the components of u_0 by arbitrary small amounts $\epsilon_i \neq 0$ as long as the constraints are not violated, so that

$$\rho_0 = u_0 + \epsilon$$

Subsequently, at the nth step of the process, let $H_n(u_n, \omega_n)$ denote an estimate of the sensitivity of the cost $J_c(\rho_n)$ with respect to ρ_n obtained over a sample path ω_n of the actual system operating under control u_n. Two sequential operations are then performed at the nth step:

1. The continuous state ρ_n is updated through

$$\rho_{n+1} = \gamma_{n+1}[\rho_n - \eta_n H_n(u_n, \omega_n)] \tag{17}$$

   where γ_{n+1} is a projection function onto the set U_c so that $\rho_{n+1} \in U_c$, depending on the nature of the set U_c, and η_n is a "step size" parameter

2. The newly determined control vector of the surrogate problem, ρ_{n+1}, is transformed into an actual feasible discrete vector of the original system through

$$u_{n+1} = f_{n+1}(\rho_{n+1}) \tag{18}$$

   where $f_{n+1} : U_c \rightarrow U$ is a mapping of feasible continuous controls to feasible discrete controls which must be appropriately selected.

One can recognize in (17) the form of a stochastic approximation algorithm[12] that generates a sequence $\{\rho_n\}$ aimed at solving (16). However, there is an additional operation (18) for generating a sequence $\{u_n\}$ which we would like to see converge to u^*, the solution of (15). It is important to note that $\{u_n\}$ corresponds to feasible realizable controls based on which one can evaluate estimates $H_n(u_n, \omega_n)$ from observable data, i.e., a sample path of the actual system under u_n (not the surrogate control ρ_n). We can therefore see that this scheme is intended to combine the advantages of a stochastic approximation type of algorithm with the ability to obtain sensitivity estimates with respect to discrete decision variables. In particular, sensitivity estimation methods for discrete parameters based on Concurrent Simulation[6] are ideally suited to meet this objective.

The cornerstones of this method are the selection of the mapping f_{n+1} in (18) and of a surrogate cost function $L_c(\rho, \omega)$ whose relationship to the actual cost must be made explicit. In addition, the estimates $H_n(u_n, \omega_n)$ necessary for the optimization scheme described above must be obtained. These are dicussed in detail in Gokbayrak and Cassandras.[7]

4.3. A Mission Planning Scenario Example

We illustrate our approach to optimal mission planning using a sample scenario shown in Figure 4. In this scenario there are $M = 16$ targets with different values protected by 16 SAM sites as shown. Our task is to plan missions at a base with two asset types (i.e., $Q = 2$), SA and WW, such that we have at our disposal $K_1 = 20$ SA and $K_2 = 8$ WW. We then seek a 32−dimensional vector $u = [u_{1,1}, u_{1,2}, \cdots, u_{16,1}, \ldots, u_{16,2}]$ to maximize a reward function of the form (15) or, if we approximate the cost-to-go $E[J^{\pi^{RO}}_{k+1}(f(x_k, u_k, w_k))]$ in (10), we can minimize $\hat{Q}(x_k, u_k)$ in (12). As shown in Figure 4, packages are assumed to fly in a straight line between base and assigned target and must, therefore, engage threats on their way to a target or during the returning part of the mission. The number of possible solutions to this problem is about 10^{18}, and that is only reflective of the combinatorial complexity aspect of it.

An added complication in this problem is the fact that there are multiple local optima. The "surrogate problem" method provides an attractive means of dealing with this difficulty because of its convergence speed. Our approach in this case is to randomize over the initial controls u_0 (equivalently, ρ_0) and seek a (possibly local) minimum corresponding to this initial point. The process is repeated for different, randomly selected, initial controls so as to seek better solutions. For deterministic problems, the best allocation seen so far is reported as the optimal. For stochastic problems, we adopt the stochastic comparison approach in Gong et al.[9] The algorithm is run from a randomly selected initial point and the cost of the corresponding final point is compared with the cost of the "best

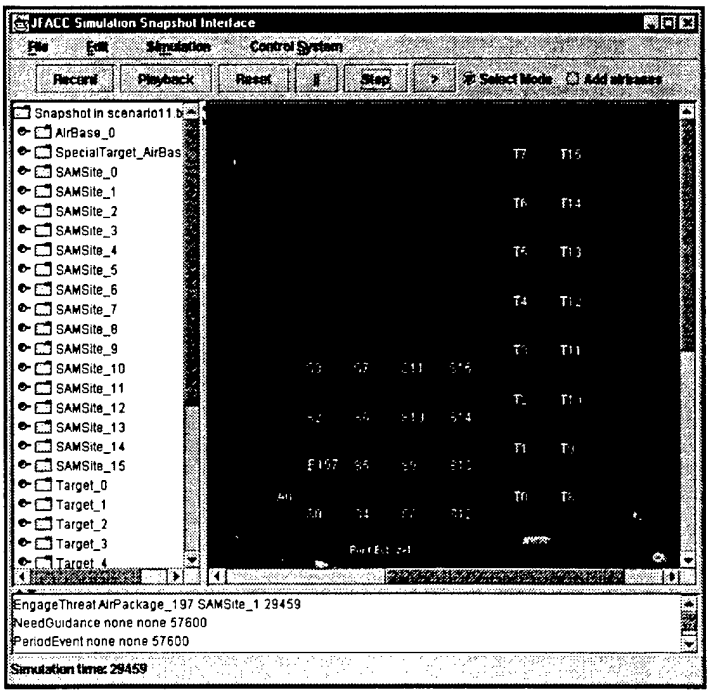


Figure 4. A mission planning scenario example

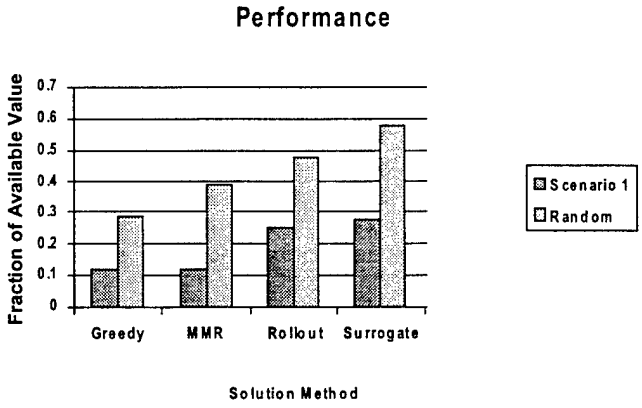


Figure 5. Performance comparison over different solution approaches (single mission wave)

point seen so far". The stochastic comparison test[9] is applied to determine the "best point seen so far" for the next run.

The surrogate problem approach was applied for a single mission wave and its performance was compared to three alternatives as shown in Figure 5. Here, "Greedy" refers to a simple greedy heuristic used also in Wohletz et al[14] in which targets are ordered on the basis of their value and missions are assigned from most to least valuable target without taking into account the number of available assets at the base. Another heuristic, labeled "MMR", uses the maximal marginal return in expected mission reward and assigns packages based on this metric, computed using an analytical model that includes detailed engagements and a limited amount of interdependencies of packages in engaging threats (therefore, it tends to be conservative). Finally, "Rollout" refers to a rollout algorithm[3] applied to

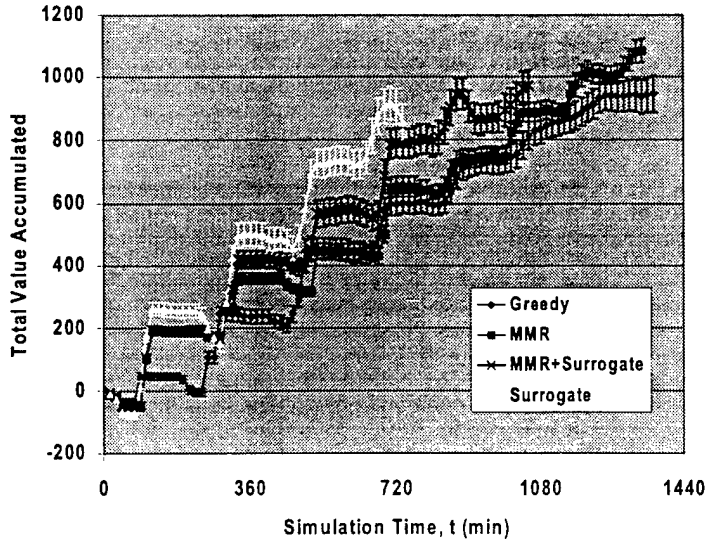


Figure 6. Total campaign value comparison over different solution approaches (multiple mission waves)

a single mission wave. Note that performance in these results is measured as the fraction of total attainable reward with respect to the total target value. In addition to the scenario described above, the results in Figure 5 include an average over a number of randomly generated scenaria.

In the case of a campaign with multiple mission waves, the surrogate problem method was applied independently from one wave to the next and the performance results are shown in Figure 6 (which includes confidence intervals). In this case, "MMR+Surrogate" refers to the use of the MMR algorithm to determine an initial point for the surrogate problem method, which accelerates its convergence. Note that this method brings the campaign to an end faster than the other approaches shown; its lack of anticipative capabilities shows in that it does not attain additional value that the MMR heuristic, for example, can at the expense of a longer campaign. This is also illustrated in Figure 7: The surrogate problem method is seen to destroy approximately the same number of targets as the other three approaches in less time, but at the expense of higher asset attrition. The "MMR+Surrogate" controller, on the other hand, can significantly reduce attrition at the expense of a somewhat longer campaign.

5. CONCLUSIONS AND FUTURE WORK

The quest for agile control in a JAO environment depends on appropriate battle space modeling and formulation of a stochastic dynamic optimization problem for mission planning. In this paper, we have described the complexities associated with these tasks and some approaches for solving optimal mission planning problems. These approaches combine approximation methods for solving notoriously hard dynamic programmimg problems with more recent advances in optimal resource allocation techniques. They all require estimating performance-related quantities under alternative controllers, which is often accomplished using discrete event simulation. Thus, part of our ongoing work involves the development of simulation tools and related efficient estimation methods.

Our ultimate goal is to develop closed-loop control and optimization methods capable of taking advantage of battle space data as they become available in real time, while also anticipating uncertain future events. Toward this goal, we are exploring new approximation methods and optimization algorithms that can incorporate simulation-based estimates. It is possible, for example, to exploit the structure of the uncertainty factors entering in this problem by extracting key features rendering estimation more efficient with little loss of accuracy. At the same time, we need to include additional features of the JAO setting into our models, such as the presence of time-critical targets or "intelligent" threats.

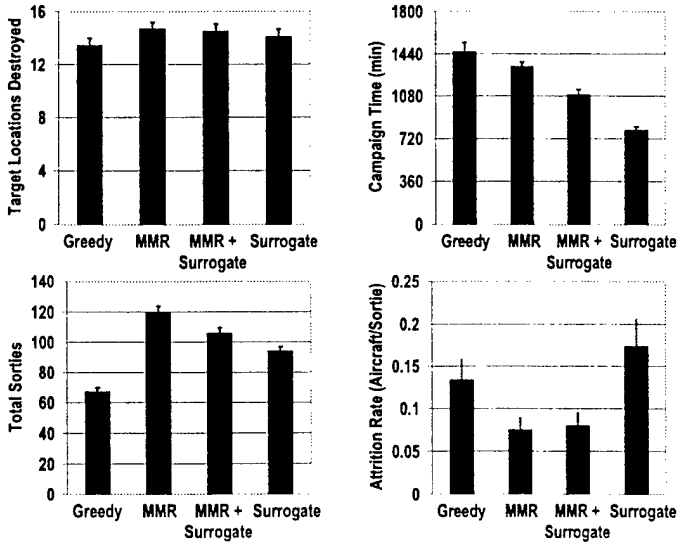


Figure 7. Performance comparison over different solution approaches (single mission wave)

ACKNOWLEDGMENTS

This work is supported in part by the Air Force Research Laboratory under contract F30602-99-C-0057 and by AFOSR under grant F49620-01-0056.

REFERENCES

1. E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines.* Wiley, New York, NY, 1989.
2. D. P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, Massachusetts, 1995.
3. D.P. Bertsekas and J.N. Tsitsiklis. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3(3):245–262, 1997.
4. C. G. Cassandras, L. Dai, and C. G. Panayiotou. Ordinal optimization for deterministic and stochastic resource allocation. *IEEE Trans. Automatic Control*, 43(7):881–900, 1998.
5. C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, 1999.
6. C. G. Cassandras and C. G. Panayiotou. Concurrent sample path analysis of discrete event systems. *Journal of Discrete Event Dynamic Systems: Theory and Applications*, 9:171–195, 1999.
7. K. Gokbayrak and C. G. Cassandras. A generalized 'surrogate problem' methodology for on-line stochastic discrete optimization. *J. of Optimization Theory and Applications.* Submitted 2001.
8. K. Gokbayrak and C. G. Cassandras. An on-line 'surrogate problem' methodology for stochastic discrete resource allocation problems. *J. of Optimization Theory and Applications*, 108(2):349–376, 2001.
9. W. B. Gong, Y. C. Ho, and W. Zhai. Stochastic comparison algorithm for discrete optimization with estimation. *Proc. of 31st IEEE Conf. on Decision and Control*, pages 795–800, 1992.
10. Y. C. Ho, R. S. Sreenivas, and P. Vakili. Ordinal optimization in DEDS. *J. of Discrete Event Dynamic Systems: Theory and Applications*, 2:61–88, 1992.
11. J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.
12. H.J. Kushner and D.S. Clark. *Stochastic Approximation for Constrained and Unconstrained Systems.* Springer-Verlag, Berlin, Germany, 1978.
13. L. Shi and S. Olafsson. Nested partitions method for global optimization. *Operations Research*, 48:390–407, 2000.

14. J.M. Wohletz, D.A. Castanon, and M.L. Curry. Closed-loop control for joint air operations. In *Proceedings of 2001 American Control Conference*, 2001. To appear.
15. D. Yan and H. Mukai. Stochastic discrete optimization. *SIAM Journal on Control and Optimization*, 30:549–612, 1992.